OTRS
Open Technology
Real Services

Documentation

OTRS 5 - Admin Manual

Build Date:

2015-05-26

OTRS 5 - Admin Manual

René Bakker, Stefan Bedorf, Michiel Beijen, Shawn Beasley, Hauke Böttcher, Jens Bothe, Udo Bretz, Martin Edenhofer, Carlos Javier García, Martin Gruner, Manuel Hecht, Christopher Kuhn, André Mindermann, Marc Nilius, Elva María Novoa, Henning Oschwald, Martha Elia Pascual, Thomas Raith, Carlos Fernando Rodríguez, Stefan Rother, Rolf Schmidt, Burchard Steinbild, Michael Thiessmeier, Daniel Zamorano.

The source code of this document can be found at github, in the repository doc-admin. Contributions are more than welcome. You can also help translating it to your language at Transifex.

Table of Contents

List of Figures

List of Tables

List of Examples

Preface

This book is intended for use by OTRS administrators. It also serves as a good reference for OTRS newbies.

The following chapters describe the installation, configuration, and administration of the OTRS software. The first third of the text describes key functionality of the software, while the remainder serves as a reference to the full set of configurable parameters.

This book continues to be a work in progress, given a moving target on new releases. We need your feedback in order to make this a high quality reference document: one that is usable, accurate, and complete. Please write to us if you find content missing in this book, if things are not explained sufficiently, or even if you see spelling mistakes, grammatical errors, or typos. Any kind of feedback is highly appreciated and should be made via our bug tracking system on http://bugs.otrs.org. Thanks in advance for your contributions!

Chapter 1. Introduction

1. Trouble Ticket Systems - The Basics

This chapter offers a brief introduction to trouble ticket systems, along with an explanation of the core concept of a trouble ticket. A quick example illustrates the advantages of using such a system.

1.1. What is a trouble ticket system, and why do you need one?

The following example describes what a trouble ticket system is, and how you might benefit from using such a system at your company.

Let's imagine that Max is a manufacturer of video recorders. Max receives many messages from customers needing help with the devices. Some days, he is unable to respond promptly or even acknowledge the messages. Some customers get impatient and write a second message with the same question. All messages containing support requests are stored in a single inbox folder. The requests are not sorted, and Max responds to the messages using a regular email program.

Since Max cannot reply fast enough to all the messages, he is assisted by the developers Joe and John in this. Joe and John use the same mail system, accessing the same inbox. They don't realize that Max often gets two identical requests from one frustrated customer. Sometimes they both end up responding separately to the same request, with the customer receiving two different answers. Furthermore, Max is unaware of the details of their responses. He is also unaware of the details of the customer problems and their resolutions, such as which problems occur with high frequency, or how much time and money he has to spend on customer support.

At a meeting, a colleague tells Max about trouble ticket systems and how they can solve Max's problems with customer support. After looking for information on the Internet, Max decides to install OTRS on a computer that is accessible from the web by both his customers and his employees. Now, the customer requests are no longer sent to Max's private inbox but to the mail account that is used for OTRS. The ticket system is connected to this mailbox and saves all requests in its database. For every new request, the system automatically generates an answer and sends it to the customer so that the customer knows that his request has arrived and will be answered soon. OTRS generates an explicit reference, the ticket number, for every single request. Customers are now happy because their requests are acknowledged and it is not necessary to send a second message with the same question. Max, John, and Joe can now log into OTRS with a simple web browser and answer the requests. Since the system locks a ticket that is answered, no message is edited twice.

Let's imagine that Mr. Smith makes a request to Max's company, and his message is processed by OTRS. John gives a brief reply to his question. But Mr. Smith has a follow-up question, which he posts via a reply to John's mail. Since John is busy, Max now answers Mr. Smith's message. The history function of OTRS allows Max to see the full sequence of communications on this request, and he responds with a more detailed reply. Mr. Smith does not know that multiple service representatives were involved in resolving his request, and he is happy with the details that arrived in Max's last reply.

Of course, this is only a short preview of the possibilities and features of trouble ticket systems. But if your company has to attend to a high volume of customer requests through emails and phone calls, and if different service representatives need to respond

at different times, a ticket system can be of great assistance. It can help streamline work flow processes, add efficiencies, and improve your overall productivity. A ticket system helps you to flexibly structure your Support or Help Desk environment. Communications between customers and service staff become more transparent. The net result is an increase in service effectiveness. And no doubt, satisfied customers will translate into better financial results for your company.

1.2. What is a trouble ticket?

A trouble ticket is similar to a medical report created for a hospital patient. When a patient first visits the hospital, a medical report is created to hold all necessary personal and medical information on him. Over multiple visits, as he is attended to by the same or additional doctors, the attending doctor updates the report by adding new information on the patient's health and the ongoing treatment. This allows any other doctors or the nursing staff to get a complete picture on the case at hand. When the patient recovers and leaves the hospital, all information from the medical report is archived and the report is closed.

Trouble ticket systems such as OTRS handle trouble tickets like normal email. The messages are saved in the system. When a customer sends a request, a new ticket is generated by the system which is comparable to a new medical report being created. The response to this new ticket is comparable to a doctor's entry in the medical report. A ticket is closed if an answer is sent back to the customer, or if the ticket is separately closed by the system. If a customer responds again on an already closed ticket, the ticket is reopened with the new information added. Every ticket is stored and archived with complete information. Since tickets are handled like normal emails, attachments and contextual annotations will also be stored with each email. In addition, information on relevant dates, employees involved, working time needed for ticket resolution, etc. are also saved. At any later stage, tickets can be sorted, and it is possible to search through and analyze all information using different filtering mechanisms.

2. OTRS Help Desk

This chapter describes the features of OTRS Help Desk (OTRS). You will find information about the hardware and software requirements for OTRS. Additionally, in this chapter you will learn how to get commercial support for OTRS, should you require it, and how to contact the community.

2.1. Basics

OTRS Help Desk (OTRS) is a web application that is installed on a web server and can be used with a web browser.

OTRS is separated into several components. The main component is the OTRS framework which contains all central functions for the application and the ticket system. It is possible to install additional applications such as OTRS::ITSM modules, integrations with Network Monitoring solutions, a knowledge base (FAQ), et cetera.

2.2. Features

OTRS has many features. The following list gives an overview of the main features included in the OTRS framework.

2.2.1. User Interface

- OTRS comes with separate, modern web interfaces for agents and customers.

- It can be used on any modern web browser, including mobile platforms and is retina ready.
- The web interface can be customized with own themes and skins.
- Powerful and customizable agent dashboard with personal ticket overviews and graphical statistics support.
- An extensible reporting engine provides various statistics and report scheduling options.
- With the ProcessManagement it is possible to define own ticket-based screens and processes (ticket workflows).
- OTRS has a built-in rights management that can be extended with fine-grained access control lists (ACLs).
- Support for more than 30 languages and different time zones.

2.2.2. Email Interface

- Support for MIME emails with attachments.
- Automatic conversion of HTML into plain text messages (increased security for sensitive content and enables faster searching).
- Incoming mail can be filtered and pre-processed with complex rules, e.g. for spam messages or Queue distribution.
- Support for PGP and S/MIME standards for key/certificate management and email processing.
- Automatic responses, configurable for every queue.
- Email notifications for agents about new tickets, follow-ups or unlocked tickets.
- It is possible to define an own Ticket identifier to recognize follow-ups, e.g. Call#, Ticket# or Request#. There are different ticket number generators (date-based, random etc.) and you can integrate your own as well. Follow-ups can also be recognized by In-Reference-To headers or external ticket numbers.

2.2.3. Tickets

- OTRS uses Tickets to gather all external an internal communication that belongs together. These tickets are organized in Queues.
- There are many different ways of looking at the tickets in a system (based on Queues, Status, Escalation etc.) in different level of detail (small/medium/preview).
- The Ticket history records all changes to a ticket.
- Tickets can be changed in many ways, such as replying, forwarding, bouncing, moving to another Queue, updating attributes (state, priority etc.), locking and accounting working time. It is possible to modify many tickets at once (bulk action).
- Pending time and escalation time / SLA management allow time-based scheduling and restrictions on tickets.
- Tickets can be linked to other tickets or other objects such as FAQ entries.
- Automatic and timed actions on tickets are possible with the "GenericAgent".

- OTRS comes with a powerful search engine that allows complex and fulltext searches on tickets.

2.2.4. System

- OTRS runs on many operating systems (Linux, Solaris, AIX, FreeBSD, OpenBSD, Mac OS 10.x) and supports several database systems for the central OTRS back-end (MySQL, PostgreSQL, Oracle, MSSQL).
- The core system can be extended by installing OTRS packages. There are many free packages (such as FAQ, OTRS::ITSM and others) as well as FeatureAddon packages that are available for service contract customers of the OTRS group.
- Integration of external back-ends for the customer data, e.g. via AD, eDirectory or OpenLDAP. Customers can authenticate via database, LDAP, HTTPAuth or Radius.
- With the GenericInterface it is easy to connect OTRS to other web services. Simple web services can be integrated without programming, complex scenarios with custom extensions. The OTRS Ticket connector allows the creation, updating and searching of tickets, via web services from a third party application.

Now let us look at the changes in recent versions of OTRS.

2.2.5. New Features of OTRS 5

2.2.5.1. Productivity

- OTRS is now optimized for use on different types and sizes of mobile devices.
- Single-select and multi-select input fields have been modernized and provide advanced searching and filtering capabilities (thanks to Dusan Vuckovic at Mühlbauer).
- Images can now be added/uploaded to the WYSIWYG editor using Copy&Paste and Drag&Drop from anywhere outside the application (in all browsers, without additional Add-On).
- Improved ticket notification system. It is now possible to configure own ticket notifications with own trigger conditions and recipients. With OTRS Business Solution™, notifications can also be delivered via SMS and/or Notification Web View. The latter is a special screen in OTRS that holds all notifications of the agent; with this OTRS can be used entirely without an email client.
- Statistics received a new graphical user interface which is much better accessible and helps to create great statistics quickly and easily.
- Additionally, statistics support the new time periods “quarter” and “half-year”
- It is now possible to group action menu items in the ticket zoom screen. Less often used items can be grouped in a submenu, improving screen usage and clarity.
- Ticket overviews can now display customer company data, thanks to Renée Bäcker.
- The ticket process TransitionAction “TicketCreate” can now create tickets without articles.

2.2.5.2. Scalability & Performance

- The new OTRS Daemon handles all asynchronous and periodic tasks and replaces all previous OTRS cron jobs. In a clustered environment the load is automatically distributed over the nodes.

- It is now possible to specify multiple readonly mirror (slave) databases for expensive computations such as statistics or fulltext searches to distribute the load among these database servers.

2.2.5.3. Security

- A new two-factor authentication layer allows added login security.
  - If entering a fixed username and password doesn't satisfy your requirements, you can now additionally use the open standard for time based one-time passwords (RfC 6238, also known as Google Authenticator).
  - After having enabled the two-factor authentication, agents and customers can add a shared secret to their preferences and immediately start logging in using one-time passwords created by a compatible method of their choice (e.g. the Android Google Authenticator app).

2.2.5.4. Working with External Systems

- A new XSLT based GenericInterface mapping module allows for arbitrarily complex user-defined data mapping.

2.2.5.5. Installation & Administration

- The new OTRS console makes working on the commandline easy and fun. All commands have a consistent interface, useful documentation and provide helpful colored output.
- Administrators can now specify a minimum log level to reduce logging volume, thanks to Renée Bäcker.
- Overview screens in the admin area now show invalid entities in gray, making it easy to focus on active elements.

2.2.6. New Features of OTRS 4

2.2.6.1. Productivity

- A new cleaner flat design has been implemented.
- Agents can now reply directly to a ticket note. The original notes body is quoted in the new note.
- Agents can now make use of templates in all screens with internal notes.
- Ticket action screens (such as note, owner etc.) now allow to do actions without always creating an article (configurable).
- New ticket overview based on "my services" that an agent can subscribe to. Notification options for new tickets and follow-ups can now be based on "my queues", "my services" or combinations of both.
- OTRS can now display tickets with thousands of articles.
- Customer online list in Dashboard now links directly to CustomerInformationCenter page for the customer.
- Agents can now persistently reorder their main menu with drag&drop.
- Agents and customers can now search tickets by attachment name.

- New Dashboard Widget for running process tickets.
- New search options for the last change time of the ticket.
- Added new screen for outgoing emails on a ticket that are not replies.

2.2.6.2. Scalability & Performance

- OTRS 4 can handle more concurrent users/requests on the same hardware, and response times for single requests are shorter as well, especially for pages with lots of data.

2.2.6.3. Working With External Systems

- The GenericInterface now also supports HTTP REST as network transport protocol.

2.2.6.4. Installation & Administration

- Postmaster filters are no longer limited to 4 match/set fields. They can now have a configurable amount of fields (default 12, up to 99).
- A new configuration option Ticket::MergeDynamicFields makes it possible to specify which dynamic fields should also be merged when a ticket is merged to another ticket.
- Added new options to check dynamic fields of type text on patterns relating to error messages (translated), if they do not match.
- Added new options to restrict dynamic fields of type date/datetime on future or past dates.
- OTRS can be configured to automatically unlock a ticket if articles are added and the owner is out of office.
- Linked tickets of a specific type (e.g. merged or removed) can now be hidden via SysConfig option.
- ACL handling has been improved, made more consistent and easier to debug.
  - Added new ACL option PossibleAdd to add items to a possible list without resetting (like Possible does).
  - Added new ACL value modifiers [Not], [NotRegExp], [Notregexp], for all ACLs parts.
- Process handling has been improved, made more consistent and easier to debug.
  - A new GUID-based entity naming scheme for the OTRS Process configuration makes it possible to safely transfer processes from one system to another without duplicating the entities.
  - Added new Transition Action to create a new ticket.
  - Added possibility to define variable Transition Action attributes based on current process ticket values.
- The possibility to schedule System Maintenance periods is available from the System Administration panel in the Admin interface.
  - A notification about an incoming System Maintenance period will be shown with some (configurable) time in advance.
  - If a System Maintenance is active, a notification about it will be shown on the Agent and Customer interface, and only admin users can log on to the system.

  - An overview screen informs admins about active sessions, which can be ended all on one click or one by one.

- Added possibility to disable sysconfig import via configuration.
- Added Apache MD5 as a new password hashing backend, thanks to Norihiro Tanaka.
- Added the possibility to restrict customer self registration by email address whitelist or blacklist, thanks to Renée Bäcker.
- Added new dashboard module that shows the output of an external command, thanks to ib.pl.

2.2.6.5. Development

- New powerful template engine based on Template::Toolkit.
- A central object manager makes creating and using global objects much easier (thanks to Moritz Lenz @ noris network).
- The OPM package format was extended to signal that a package has been merged into another package, allowing the package manager to correctly handle this situation on package installation or update.
- Caching was centralized in one global cache object which also performs in-memory caching for all data.
- Added cache benchmark script, thanks to ib.pl.

2.2.7. New Features of OTRS 3.3

2.2.7.1. Productivity

- Dashboard ticket lists and regular ticket overviews can now be filtered by eligible ticket columns, and the shown columns are configurable.
- Ticket medium and preview overviews are now sortable.
- Added a calendar widget for the dashboard that can show tickets as events.
- Added new dashboard widget that shows in a matrix form the number of tickets per state and per queue.
- Agents can now mark important articles.
- A new tree selection widget makes working with tree data (queues, services etc.) much faster and easier.
- Added support to search relative dates (e.g. more than 1 month ago) in Date and Date/Time dynamic fields.
- It is now possible to specify templates (previously "standard responses") also for creation of new tickets and for ticket forwarding.
- The list of available processes can now be filtered by ACLs.
- Added support to initiate processes from Customer Interface.
- In many places text is not shortened any more by a fixed number of characters ("Queue1..."), but instead by available screen estate. This makes it possible to see more information at once.

- OTRS is now Retina-ready. Images have been adapted to match the higher resolutions and most of the image icons have been replaced by font characters from the FontAwesome webfont.
- Added new feature "management dashboard". This makes it possible to display statistic charts in the dashboard. Please note that IE8 does not support this feature.

2.2.7.2. Working With External Systems

- OTRS can now use multiple customer company databases, thanks to Cyrille @ belnet-ict.
- OTRS can now automatically store customer user data in ticket dynamic fields for permanent storage in the ticket. This can be useful for reporting.
- OTRS is now able correctly assign incoming emails to existing tickets based on ticket numbers from external systems.
- OTRS can now fetch email also over POP3/TLS connections.

2.2.7.3. Installation & Administration

- Web Installer now can setup OTRS on PostgreSQL, Oracle and SQL Server databases in addition to MySQL.
- OTRS now has full support for MySQL 5.6.
- Generic agent jobs can now be executed for configured ticket events.
- The new graphical ACL editor makes ACL editing easier.
- Postmaster filters can now use negated filter conditions, thanks to Renée Bäcker.
- Postmaster filters can now specify relative pending dates and Owner / Responsible for new tickets based on incoming email data.
- Customer and Agent passwords now can be encrypted using the strong bcrypt algorithm, which is better than SHA.
- Many icons now use an icon font which makes it much easier to create custom skins with different base colors. This also improves overall performance through smaller amount of (image) files to load.

2.3. Hardware and Software Requirements

OTRS can be installed on many different operating systems. OTRS can run on linux and on other unix derivates (e.g. OpenBSD or FreeBSD). OTRS does not have excessive hardware requirements. We recommend using a machine with at least a 2 GHz Xeon or comparable CPU, 2 GB RAM, and a 160 GB hard drive for a small setup.

To run OTRS, you'll also need to use a web server and a database server. Apart from that, you should install perl and/or install some additional perl modules on the OTRS machine. The web server and Perl must be installed on the same machine as OTRS. The database back-end may be installed locally or on another host.

For the web server, we recommend using the Apache HTTP Server, because its module mod_perl greatly improves the performance of OTRS. Apart from that, OTRS should run on any web server that can execute Perl scripts.

You can deploy OTRS on different databases. You can choose between MySQL, PostgreSQL or Oracle. If you use MySQL or PostgreSQL you have the advantage that the database and some system settings can be configured during the installation, through a web front-end.

For Perl, you will need some additional modules which can be installed either with the Perl shell and CPAN, or via the package manager of your operating system (rpm, yast, apt-get).

Software requirements

2.3.1. Perl support

- Perl 5.10 or higher

2.3.2. Web server support

- Apache2 + mod_perl2 or higher (recommended)
- Webserver with CGI support (CGI is not recommended)

2.3.3. Database support

- MySQL 5.0 or higher
- MariaDB
- PostgreSQL 8.4 or higher
- Oracle 10g or higher

The section in the manual about installation of Perl modules describes in more detail how you can set up those which are needed for OTRS.

If you install a binary package of OTRS, which was built for your operating system (rpm), either the package contains all Perl modules needed or the package manager of your system should take care of the dependencies of the Perl modules needed.

2.3.4. Web browser support

To use OTRS, you'll be OK if you use a modern browser with JavaScript support enabled. These browsers are not supported:

- Internet Explorer before version 10
- Firefox before version 10
- Safari before version 5

We recommend keeping your browser up-to-date. JavaScript and rendering performance in newer versions is always improved. Dramatic performance issues can be seen in larger systems when using older versions. We are happy to consult you on that matter.

2.4. Community

OTRS has a large user community. Users and developers discuss OTRS and exchange information on related issues through the mailing-lists. You can use the mailing lists to discuss installation, configuration, usage, localization and development of OTRS. You can report software bugs in our bug tracking system.

The homepage of the OTRS community is: http://www.otrs.com/open-source/.

2.5. Professional Services for OTRS

Our OTRS Business Solution™ offers you best professional support from the OTRS team, reliable OTRS security and regular free updates as well as an exclusive set of additional

Business Features that you can flexibly activate or deactivate according to different deployment scenarios.

The OTRS Group offers specific training programs in different countries. You can either participate in one of our public OTRS Administrator trainings which take place regularly, or benefit from an inhouse training that covers all the specific needs of your company.

Chapter 2. Installation

This chapter describes the installation and basic configuration of the central OTRS framework. It covers information on installing OTRS from source, or with a binary package such as an RPM.

Topics covered here include configuration of the web and database servers, the interface between OTRS and the database, the installation of additional Perl modules, setting proper access rights for OTRS, setting up the cron jobs for OTRS, and some basic settings in the OTRS configuration files.

Follow the detailed steps in this chapter to install OTRS on your server. You can then use its web interface to login and administer the system.

1. The Simple Way - Installation of Pre-Built Packages

If available for your platform you should use pre-built packages to install OTRS, since it is the simplest and most convenient method. You can find them in the download area at www.otrs.com. The following sections describe the installation of OTRS with a pre-built or binary package on SUSE and Red Hat systems. Only if you are unable to use the pre-built packages for some reason should you follow the manual process.

1.1. Installing the RPM on a SUSE Linux server

This section describes the installation of our RPM package on a SUSE Linux server.

1.1.1. Preparing the database for OTRS

You can use OTRS using different database back-ends: MySQL, PostgreSQL or Oracle. The most popular database to deploy OTRS on is MySQL. This chapter shows the steps you need to take to configure MySQL on a SUSE-based server. Of course you can install the database on a dedicated database server if needed for scalability or other purposes.

Note

If you follow this chapter on openSUSE 12.3 and up you'll actually not install MySQL but MariaDB instead, a MySQL compatible fork of the MySQL code. This is no problem, it will work just as well (and even a little better at some points).

Install MySQL by executing the following command as root:

```
linux:~ # zypper install mysql perl-DBD-mysql
```

This will install MySQL with the default options on your system. You'll need to change the defaults in order to make it suitable for OTRS. With a text editor open the file `/etc/my.cnf` and add following lines under the [mysqld] section:

```
max_allowed_packet   = 20M
query_cache_size     = 32M
innodb_log_file_size = 256M
```

Now execute **systemctl restart mysql.service** to re-start the database server and activate these changes. Then run **/usr/bin/mysql_secure_installation** and follow the on-screen instructions to set a database root password, remove anonymous access and remove the test database. Lastly, run **systemctl enable mysql.service** in order to make sure MySQL is automatically started at server startup time.

1.1.2. Installing OTRS

Install OTRS with via the command line using **zypper**. This will also pull in some dependencies such as the Apache web server and some Perl modules. Make sure you copied the OTRS RPM file to the current directory.

```
otrs-sles:~ # zypper install otrs*.rpm
....
Retrieving package otrs-x.x.x-01.noarch (1/26), 17.5 MiB (74.3 MiB unpacked)
Installing: otrs-x.x.x-01 [done]
Additional rpm output:
Check OTRS user ... otrs added.

...

otrs-sles:~ #
```

Now restart Apache with the command **systemctl restart apache2.service** to load the configuration changes for OTRS.

1.1.3. Installation of additional perl modules

OTRS needs more modules than can be installed via the package manager per default. You can post-install them manually. Running the `otrs.CheckModules.pl` script located at `/opt/otrs/bin/` will let you know which modules are missing, and must or can be installed. Optional modules may include those needed for communication with MDAs via IMAP(S) or gernerating PDF output.

On SLES you shoud add an external repository in order to get missing modules. Choose the repository needed for your OS version from here: http://download.opensuse.org/repositories/devel:/languages:/perl/ . As an example, the repository for SLES 11 SP 3 would be added like this:

```
zypper ar -f -n perl http://download.opensuse.org/repositories/devel:/languages:/perl/
SLE_11_SP3 Perl
```

On openSUSE 12.3 the extra repository is only needed for the Mail::IMAPClient module, which you'd only need if you need to collect mails from an IMAP server secured with TLS. The corresponding line would look like this:

```
zypper ar -f -n perl http://download.opensuse.org/repositories/devel:/languages:/perl/
openSUSE_12.3/ Perl
```

The first time you use zypper after you added this repository, you will be prompted to add its key. Now you can install missing modules like below.

```
otrs-sles:/opt/otrs # zypper install -y "perl(YAML::LibYAML)"
Refreshing service 'susecloud'.
Retrieving repository 'perl' metadata [\]
```

```
New repository or package signing key received:
Key ID: DCCA98DDDCEF338C
Key Name: devel:languages:perl OBS Project &lt;devel:languages:perl@build.opensuse.org&gt;
Key Fingerprint: 36F0AC0BCA9D8AF2871703C5DCCA98DDDCEF338C
Key Created: Wed Oct 10 22:04:18 2012
Key Expires: Fri Dec 19 22:04:18 2014
Repository: perl

Do you want to reject the key, trust temporarily, or trust always? [r/t/a/?] (r): a
Retrieving repository 'perl' metadata [done]
Building repository 'perl' cache [done]
Loading repository data...
Reading installed packages...
'perl(YAML::LibYAML)' not found in package names. Trying capabilities.
Resolving package dependencies...

The following NEW package is going to be installed:
  perl-YAML-LibYAML

The following package is not supported by its vendor:
  perl-YAML-LibYAML

Retrieving package perl-YAML-LibYAML-0.38-12.4.x86_64 (1/1), 75.0 KiB (196.0 KiB unpacked)
Retrieving: perl-YAML-LibYAML-0.38-12.4.x86_64.rpm [done (55.7 KiB/s)]
Installing: perl-YAML-LibYAML-0.38-12.4 [done]
```

The next step is to configure OTRS using the web installer, as described in this section.

Now you can start the OTRS daemon and activate corresponding watchdog cron job (this must be done by the `otrs` user):

```
shell> /opt/otrs/bin/otrs.Daemon.pl start
shell> /opt/otrs/bin/Cron.sh start
```

That's it, congratulations!

1.2. Installing OTRS on a Red Hat Enterprise Linux or CentOS system

This section describes the installation of our RPM package on a Red Hat Enterprise Linux (RHEL) or CentOS server.

1.2.1. Preparation: Disable SELinux

Note

If your system uses SELinux, you should disable it, otherwise OTRS will not work correctly.

Here's how to disable SELinux for RHEL/CentOS/Fedora:

- Configure SELINUX=disabled in the /etc/selinux/config file:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
```

```
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

- Reboot your system. After reboot, confirm that the getenforce command returns `Disabled`:

```
shell> getenforce
Disabled
```

1.2.2. Preparing the database for OTRS

You can use OTRS using different database back-ends: MySQL, PostgreSQL or Oracle. The most popular database to deploy OTRS on is MySQL. This chapter shows the steps you need to take to configure MySQL on a RHEL-based server. Of course you can install the database on a dedicated database server if needed for scalability or other purposes.

Install MySQL (or MariaDB) by executing the following command as root:

```
shell> yum -y install mariadb-server
```

This will install MySQL with the default options on your system. You'll need to change the defaults in order to make it suitable for OTRS. With a text editor create a new file `/etc/my.cnf.d/zotrs.cnf` with the following content:

```
[mysqld]
max_allowed_packet   = 20M
query_cache_size     = 32M
innodb_log_file_size = 256M
```

Now execute **systemctl start mariadb** to re-start the database server and activate these changes. Then run **/usr/bin/mysql_secure_installation** and follow the on-screen instructions to set a database root password, remove anonymous access and remove the test database.

1.2.3. Installing OTRS

Install OTRS with via the command line using **yum**. This will also pull in some dependencies such as the Apache web server and some Perl modules. Make sure you copied the OTRS RPM file to the current directory.

```
shell> yum install --nogpgcheck otrs-x.x.*.rpm
...
Dependencies Resolved

================================================================================
 Package                    Arch   Version                Repository       Size
================================================================================
Installing:
 otrs                       noarch x.x.x-01               /otrs-x.x.x-01.noarch
                                                                            74 M
Installing for dependencies:
 apr                        x86_64 1.3.9-5.el6_2          updates         123 k
 ...
 procmail                   x86_64 3.22-25.1.el6          base            163 k

Transaction Summary
================================================================================
```

```
Install      26 Package(s)

Total size: 80 M
Total download size: 6.0 M
Installed size: 88 M
Downloading Packages:
(1/25): apr-1.3.9-5.el6_2.x86_64.rpm                       | 123 kB     00:00
...
(25/25): procmail-3.22-25.1.el6.x86_64.rpm                 | 163 kB     00:00
--------------------------------------------------------------------------------
Total                                           887 kB/s | 6.0 MB     00:06
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : apr-1.3.9-5.el6_2.x86_64                                     1/26
  ...
  Installing : otrs-x.x.x-01.noarch                                        26/26
Check OTRS user ... otrs added.

...

shell>
```

Now restart Apache with the command **systemctl restart httpd.service** to load the configuration changes for OTRS.

1.2.4. Installation of additional perl modules

OTRS needs some more modules than can be installed by the RPM. You can post-install them manually. You can check what modules you are missing by running the `bin/otrs.CheckModules.pl` script located in the `/opt/otrs` directory. Some modules are only needed for optional functionality, such as communication with IMAP(S) servers or PDF generation. On Red Hat or CentOS we recommend installing these modules from the EPEL repository, a repository maintained by the Fedora project, which provides high quality packages for RHEL and derivatives. Check for more information the EPEL web site.

If you're on RHEL 7 or CentOS 7, you can get the latest package for EPEL from this site. You can add this repository to yum it in one go by copying the RPM URL you find on this page and executing this command:

```
shell> yum -y install http://download.fedoraproject.org/pub/epel/7/x86_64/e/epel-
release-7-8.noarch.rpm

...

Installed:
  epel-release.noarch 0:7-8

Complete!
```

The first time you use yum after you added this repository, you will be prompted to add its key. Now you can install missing modules like below.

```
shell> yum -y install "perl(Text::CSV_XS)"

...

Installed:
  perl-Text-CSV_XS.x86_64 0:0.85-1.el6

Complete!
shell>
```

The next step is to configure OTRS using the web installer, as described in this section.

Now you can start the OTRS daemon and activate corresponding watchdog cron job (this must be done by the `otrs` user):

```
shell> /opt/otrs/bin/otrs.Daemon.pl start
shell> /opt/otrs/bin/Cron.sh start
```

That's it, congratulations!

1.2.5. Installation of Oracle database driver on Red Hat / CentOS

If you want to deploy OTRS on an Oracle database, you'll need to compile and install the DBD::Oracle database driver. This is slightly more complicated than installing any of the other packages; this is because Oracle is a proprietary database and Red Hat nor the CentOS project are allowed to distribute drivers in their RPM repositories.

First of all, we'd need to install gcc, make and CPAN so we can compile and install the driver. Below you see the command on CentOS; on other versions it might look a little different.

```
shell> yum -y install gcc make "perl(CPAN)"
```

The next step is to obtain and install the database client. For this you would need to sign up for a free account at the Oracle website. You can download the drivers from this page: http://www.oracle.com/technetwork/database/features/instant-client/index-097480.html Please choose the Linux x86 or x86-64 version corresponding to the architecture of your system. You can check this with the **uname -i**. It is either 'x86_64' for x86-64 or 'i386' for x86. You should download the packages 'Instant Client Package - Basic', 'Instant Client Package - SQL*Plus', and 'Instant Client Package - SDK'. Save them to a location on your disk. Now as the root user you can install the packages using the following command:

```
shell> yum install oracle-instantclient*
```

After this you should set two environment variables and compile the DBD::Oracle driver. Again, perform these tasks as the root user. The steps are outlined below. Please note that for briefness some lines outputted by the commands have been removed.

```
shell> export ORACLE_HOME=/usr/lib/oracle/11.2/client64
shell> export LD_LIBRARY_PATH=$ORACLE_HOME/lib
shell> cpan
cpan[1]> look DBD::Oracle
...
Fetching with LWP:
  http://www.perl.org/CPAN/authors/id/P/PY/PYTHIAN/CHECKSUMS
Checksum for /root/.cpan/sources/authors/id/P/PY/PYTHIAN/DBD-Oracle-1.62.tar.gz ok
Scanning cache /root/.cpan/build for sizes
DONE
...
Working directory is /root/.cpan/build/DBD-Oracle-1.62-ZH6LNy
[root@localhost DBD-Oracle-1.62-ZH6LNy]# perl Makefile.PL
...
[root@localhost DBD-Oracle-1.62-ZH6LNy]# make
...
[root@localhost DBD-Oracle-1.62-ZH6LNy]# make install
```

```
...
cpan[2]> exit
Terminal does not support GetHistory.
Lockfile removed.
```

Now you should edit the file `Kernel/Config.pm` to provide ORACLE_HOME. The next step is to configure OTRS using the web installer, as described in this section.

1.3. Installing OTRS on a Debian or Ubuntu system

Important

Please install OTRS from source, and do not use the OTRS packages that Debian/Ubuntu provides.

The installation of required Perl modules is easier if you use the available packages:

```
apt-get install libapache2-mod-perl2 libdbd-mysql-perl libtimedate-perl libnet-dns-perl
 libnet-ldap-perl \
    libio-socket-ssl-perl libpdf-api2-perl libdbd-mysql-perl libsoap-lite-perl libtext-csv-
xs-perl \
    libjson-xs-perl libapache-dbi-perl libxml-libxml-perl libxml-libxslt-perl libyaml-perl \
    libarchive-zip-perl libcrypt-eksblowfish-perl libencode-hanextra-perl libmail-
imapclient-perl \
    libtemplate-perl
```

2. Installation From Source (Linux, Unix)

2.1. Preparation: Disable SELinux

Note

If your system uses SELinux, you should disable it, otherwise OTRS will not work correctly.

Here's how to disable SELinux for RHEL/CentOS/Fedora:

- Configure SELINUX=disabled in the /etc/selinux/config file:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

- Reboot your system. After reboot, confirm that the getenforce command returns `Disabled`:

```
shell> getenforce
Disabled
```

2.2. Step 1: Install .tar.gz

If you want to install OTRS from source, first download the source archive as .tar.gz, .tar.bz2, or .zip file from https://www.otrs.com/download-open-source-help-desk-software-otrs-free/

Unpack the archive (for example, using **tar**) into the directory `/opt`, and rename the directory from otrs-x.x.x to otrs (see Script below).

```
shell> tar xzf /tmp/otrs-x.x.x.tar.gz
shell> mv otrs-x.x.x /opt/otrs
```

2.3. Step 2: Install Additional Perl Modules

Use the following script to get an overview of all installed and required CPAN modules.

```
shell> perl /opt/otrs/bin/otrs.CheckModules.pl
o CGI............................ok (v3.60)
o Crypt::PasswdMD5................ok (v1.3)
o Crypt::SSLeay...................Not installed! (Optional - Required for Generic Interface
 SOAP SSL connections.)
o CSS::Minifier...................ok (v0.01)
o Date::Format....................ok (v2.22)
o Date::Pcalc.....................ok (v1.2)
...
```

Note

Please note that OTRS requires a working Perl installation with all "core" modules such as the module `version`. These modules are not explicitly checked by the script. You may need to install a `perl-core` package on some systems like RHEL that do not install the Perl core packages by default.

To install missing Perl modules, you can:

2.3.1. a) Install the packages via the package manager of your Linux distribution

- For Red Hat, CentOS, Fedora or compatible systems:

```
shell> yum install "perl(Digest::MD5)"
```

- For SUSE Linux Enterprise Server, openSUSE or compatible systems: first determine the name of the package the module is shipped in. Usually the package for My::Module would be called "perl-My-Module".

```
shell> zypper search Digest::MD5
```

Then install:

```
```

```
shell> zypper install perl-Digest-MD5
```

- For Debian, Ubuntu or compatible systems first determine the name of the package the module is shipped in. Usually the package for My::Module would be called "libmy-module-perl".

  ```
  shell> apt-cache search Digest::MD5
  ```

  Then install:

  ```
  shell> apt-get install libdigest-md5-perl
  ```

  Please note that it might be that you can't find all modules or their required versions in your distribution repository, in that case you might choose to install those modules via CPAN (see below).

2.3.2. b) Install the required modules via the CPAN shell

Note that when you're on Linux you should run CPAN as your superuser account because the modules should be accessible both by the OTRS account and the account under which the web server is running.

```
shell> perl -MCPAN -e shell;
...
install Digest::MD5
install Crypt::PasswdMD5
...
```

Any optional modules listed by the script should be installed depending on the special requirements of the target system.

2.4. Step 3: Create OTRS User

Create user:

```
shell> useradd -d /opt/otrs -c 'OTRS user' otrs
```

Add user to webserver group (if the webserver is not running as the OTRS user):

```
shell> usermod -G www otrs
(SUSE=www, Red Hat/CentOS/Fedora=apache, Debian/Ubuntu=www-data)
```

2.5. Step 4: Activate Default Config Files

There are two OTRS config files bundled in $OTRS_HOME/Kernel/*.dist and $OTRS_HOME/Kernel/Config/*.dist. You must activate them by copying them without the ".dist" filename extension.

```
shell> cd /opt/otrs/
shell> cp Kernel/Config.pm.dist Kernel/Config.pm
```

2.6. Step 5: Check if all needed modules are installed

```
shell> perl -cw /opt/otrs/bin/cgi-bin/index.pl
/opt/otrs/bin/cgi-bin/index.pl syntax OK

shell> perl -cw /opt/otrs/bin/cgi-bin/customer.pl
/opt/otrs/bin/cgi-bin/customer.pl syntax OK

shell> perl -cw /opt/otrs/bin/otrs.Console.pl
/opt/otrs/bin/otrs.Console.pl syntax OK
```

"syntax OK" tells you all mandatory Perl modules are installed.

2.7. Step 6: Configuring the Apache web server

First of all, you should install the Apache2 web server and mod_perl; you'd typically do this from your systems package manager. Below you'll find the commands needed to set up Apache on the most popular Linux distributions.

```
# rhel / centos:
shell> yum install httpd mod_perl

# suse:
shell> zypper install apache2-mod_perl

# debian/ubuntu:
shell> apt-get install apache2 libapache2-mod-perl2
```

Most Apache installations have a `conf.d` directory included. On Linux systems you can usually find this directory under `/etc/apache` or `/etc/apache2`. Log in as root, change to the `conf.d` directory and link the appropriate template in `/opt/otrs/scripts/apache2-httpd.include.conf` to a file called `zzz_otrs.conf` in the Apache configuration directory (to make sure it is loaded after the other configurations).

OTRS requires a few Apache modules to be active for optimal operation. On most platforms you can make sure they are active via the tool a2enmod.

```
shell> a2enmod perl
shell> a2enmod version
shell> a2enmod deflate
shell> a2enmod filter
shell> a2enmod headers
```

Now you can restart your web server to load the new configuration settings. On most systems you can do that with the command **systemctl restart apache2.service**.

2.8. Step 7: File Permissions

File permissions need to be adjusted to allow OTRS to read and write files:

`otrs.SetPermissions.pl` [--otrs-user= OTRS user, defaults to 'otrs'] { --web-group= group of the web server user}

For example:

- Web server which runs as the OTRS user:

```
shell> bin/otrs.SetPermissions.pl --web-user=otrs
```

- Webserver with wwwrun user (e. g. SUSE):

```
shell> bin/otrs.SetPermissions.pl --web-group=wwwrun
```

- Webserver with apache user (e. g. Red Hat, CentOS):

```
shell> bin/otrs.SetPermissions.pl --web-group=apache
```

- Webserver with www-data user (e. g. Debian, Ubuntu):

```
shell> bin/otrs.SetPermissions.pl --web-group=www-data
```

2.9. Step 8: Database Setup and Basic System Configuration

Please use the web installer at http://yourhost/otrs/installer.pl (replace "yourhost" with your OTRS hostname) to setup your database and basic system settings such as email accounts.

Note

The following configuration settings are recommended for MySQL setups. Please add the following lines to `/etc/my.cnf` under the [mysqld] section:

```
max_allowed_packet   = 20M
query_cache_size     = 32M
innodb_log_file_size = 256M
```

2.10. Step 9: First login

Now you are ready to login to your system at http://yourhost/otrs/index.pl with the credentials you configured in the web installer (User: root@localhost).

With this step, the basic system setup is finished.

2.11. Step 10: Start the OTRS Daemon

The new OTRS daemon is responsible for handling any asynchronous and recurring tasks in OTRS. What has been in cron file definitions previously is now handled by the OTRS daemon, which is now required to operate OTRS. The daemon also handles all GenericAgent jobs and must be started from the `otrs` user.

```
shell> /opt/otrs/bin/otrs.Daemon.pl start
```

2.12. Step 11: Cron jobs for the OTRS user

There are two default OTRS cron files in `/opt/otrs/var/cron/*.dist`, and their purpose is to make sure that the OTRS Daemon is running. They need to be be activated by copying them without the ".dist" filename extension.

```
shell> cd /opt/otrs/var/cron
shell> for foo in *.dist; do cp $foo `basename $foo .dist`; done
```

To schedule these cron jobs on your system, you can use the script `Cron.sh` with the `otrs` user.

```
shell> /opt/otrs/bin/Cron.sh start
```

Stopping the cron jobs is also possible (useful for maintenance):

```
shell> /opt/otrs/bin/Cron.sh stop
```

2.13. Step 12: Setup bash autocompletion (optional)

All regular OTRS commandline operations happen via the otrs Console interface `bin/otrs.Console.pl`. This provides an auto completion for the bash shell which makes finding the right command and options much easier.

You can activate the bash autocompletion by installing the package `bash-completion`. It will automatically detect and load the file `/opt/otrs/.bash_completion` for the `otrs` user.

After restarting your shell, you can just type `bin/otrs.Console.pl` followed by TAB, and it will list all available commands. If you type a few characters of the command name, TAB will show all matching commands. After typing a complete command, all possible options and arguments will be shown by pressing TAB.

2.14. Step 13: Further Information

We advise you to read the OTRS performance tuning chapter.

If you encounter problems with the installation, you can send a message to our mailing list otrs@otrs.org (http://lists.otrs.org/).

You can also ask the OTRS Group to either help you in planning or deploying OTRS, or review your installed OTRS system. Our professional services are designed to help you deploy OTRS faster and to get the most benefit out of OTRS.

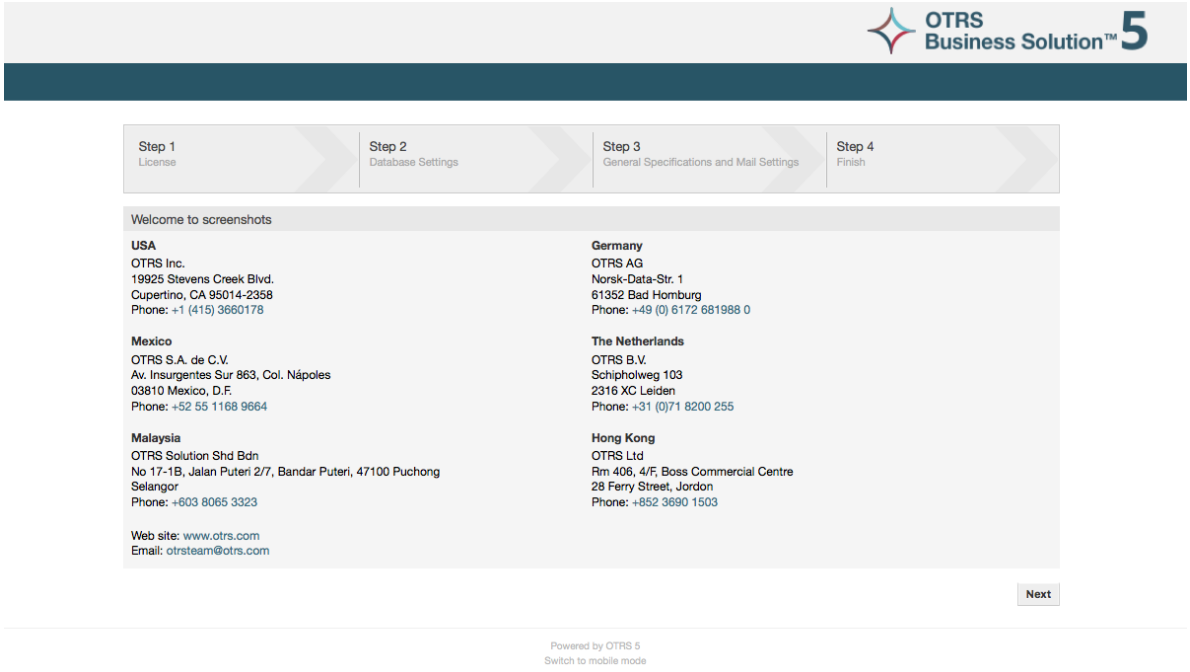
3. Using the Web Installer

You can use the OTRS Web Installer, after you installed the OTRS software, to set up and configure the OTRS database. The Web Installer is a web page you can visit in your browser. The URL for the web installer is http://localhost/otrs/installer.pl .

When the web installer starts, please follow the following steps to setup your system:

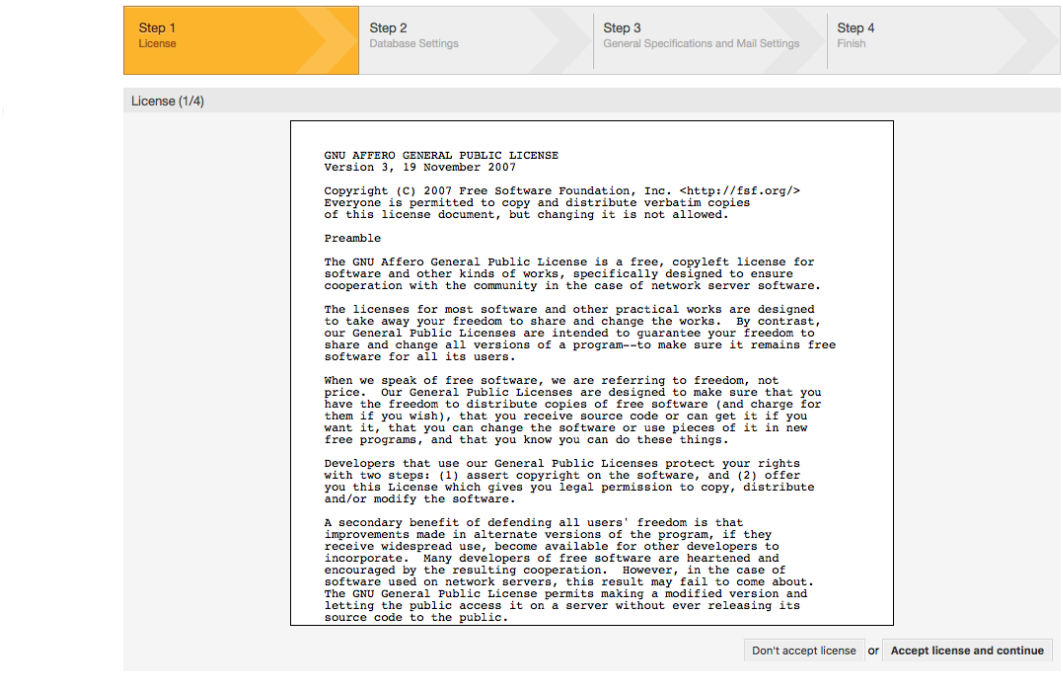
1. Check out the information about the OTRS offices and click on 'Next' to continue (see figure below).

Figure 2.1. Welcome screen



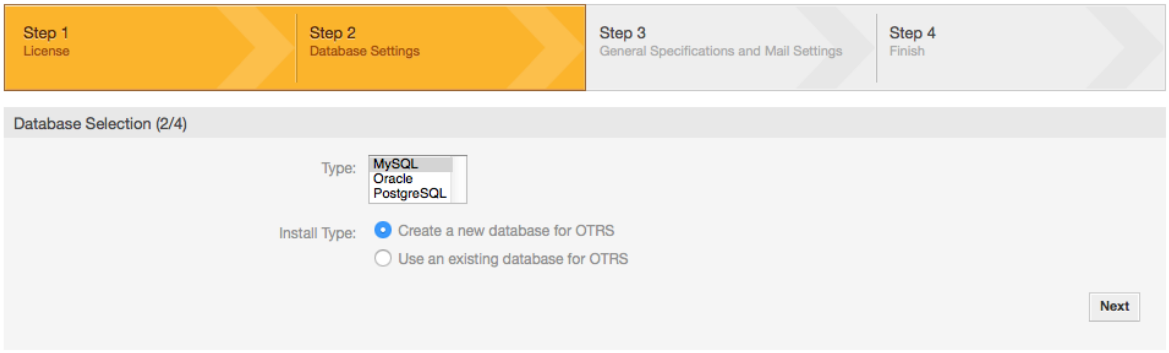
2. Read the GNU Affero General Public License (see figure below) and accept it, by clicking the corresponding button at the bottom of the page.

Figure 2.2. GNU Affero General Public License



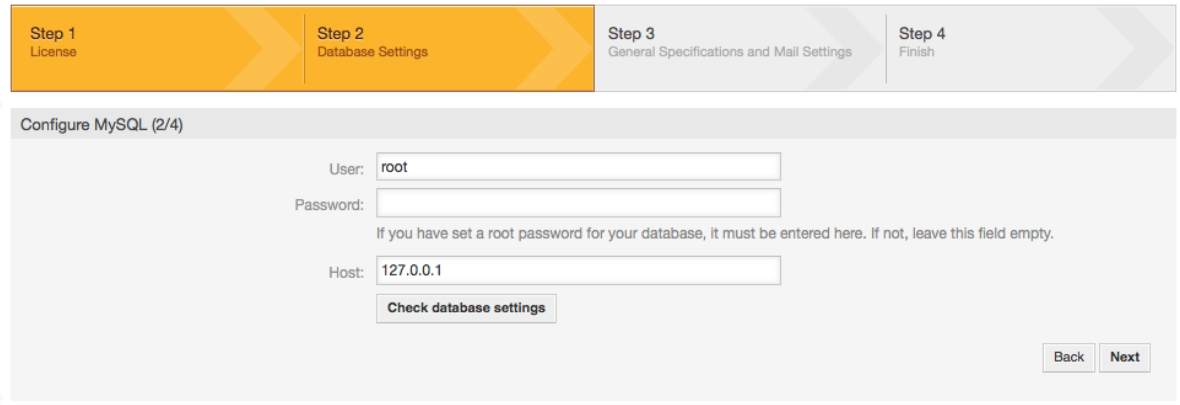
3. Choose the database that you want to use with OTRS. If you choose MySQL or PostgreSQL as a database, you can also select here if you want the web installer to create a database for you or if your database administrator has already created an empty database for you that you would like to use. After that, click the 'Next' button (see figure below).

Figure 2.3. Database Selection



4. Depending of the database you chose and if you wanted the web installer to create a database or use an existing one in the previous step, this screen might differ a little. Enter the credentials for the database in this screen.

Figure 2.4. Database credentials

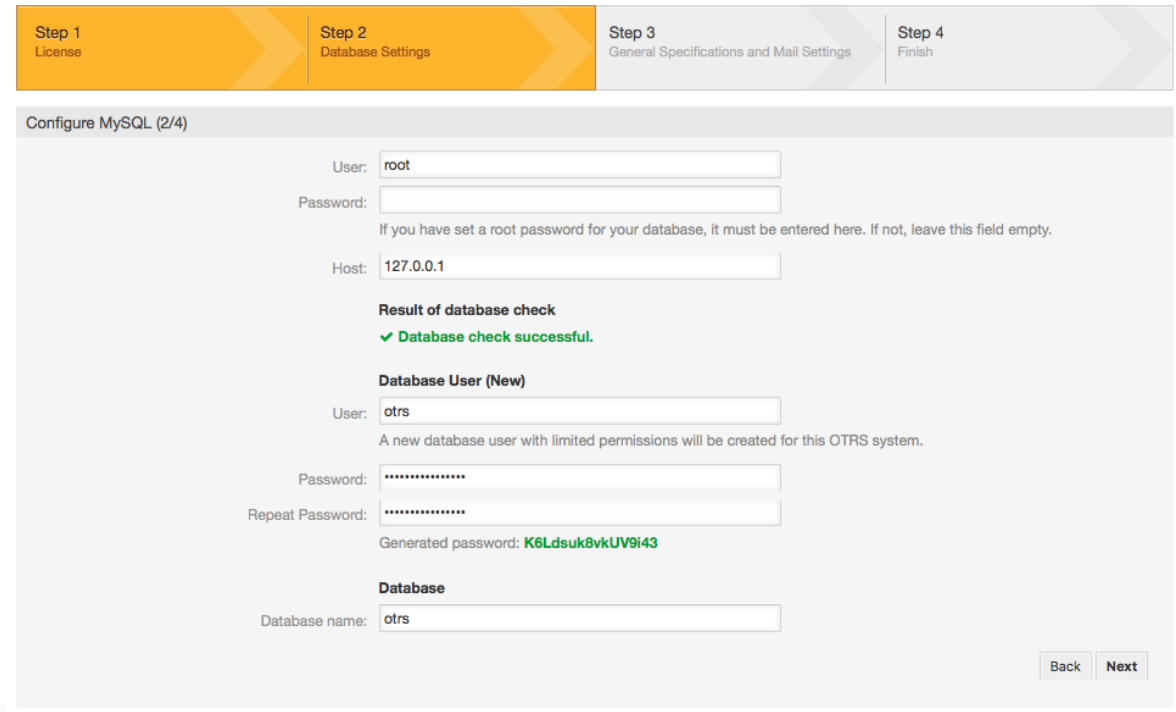


5. Create a new database user, choose a name for the database and click on 'Next' (see figure below).

Warning

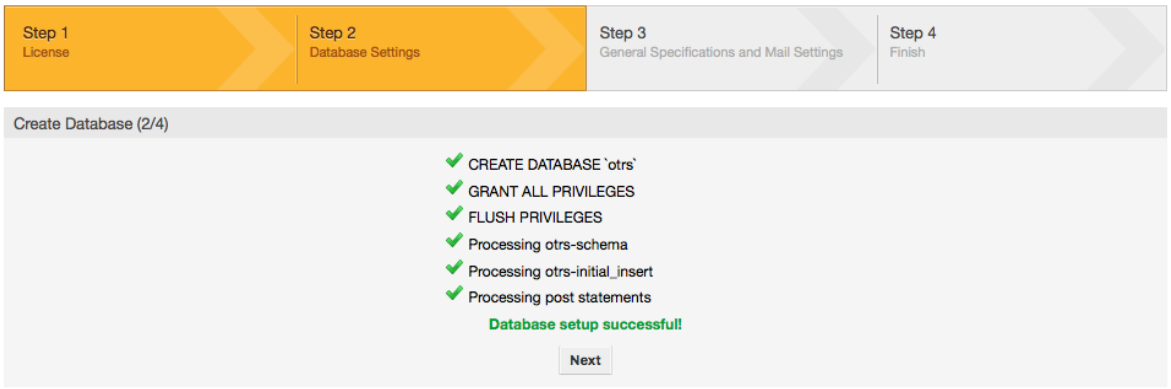
OTRS will generate a strong password for you. It's possible to enter your own password if you prefer this. The password will be written to the configuration file `Kernel/Config.pm` so there is no need to remember this password.

Figure 2.5. Database settings



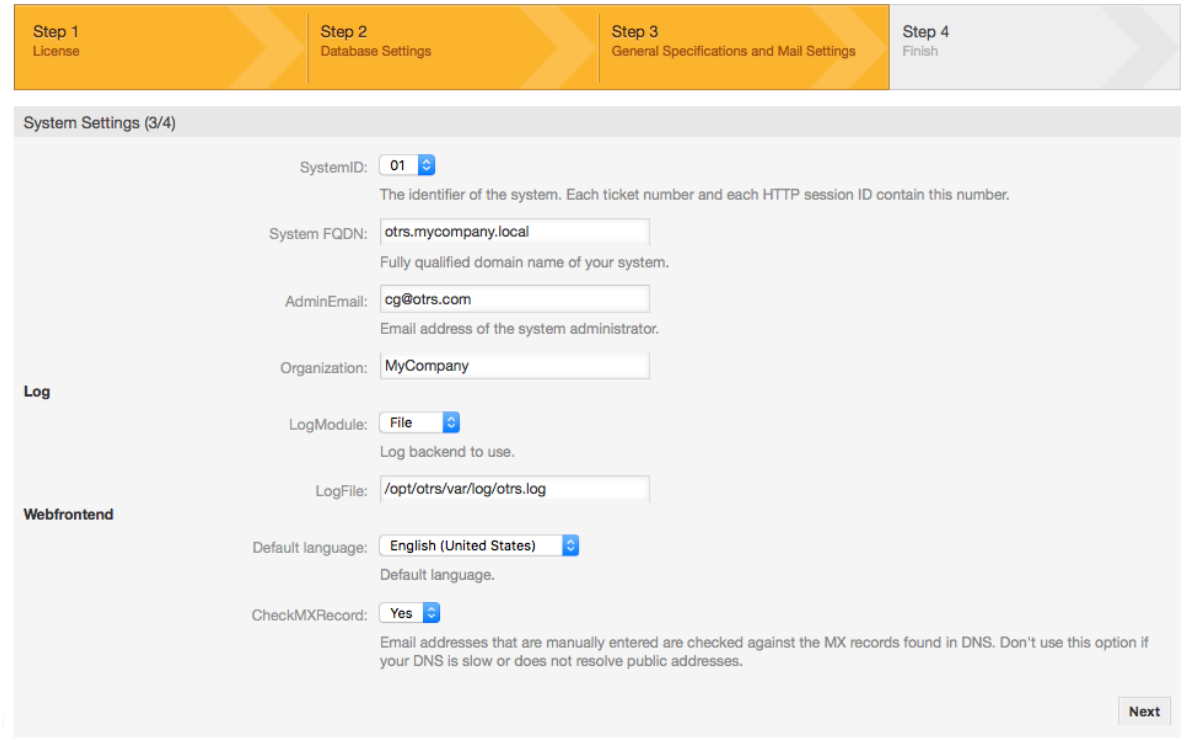
6. The database will be created if needed, and populated, as shown in this image. Click 'Next' to go to the next screen.

Figure 2.6. Successful database setup



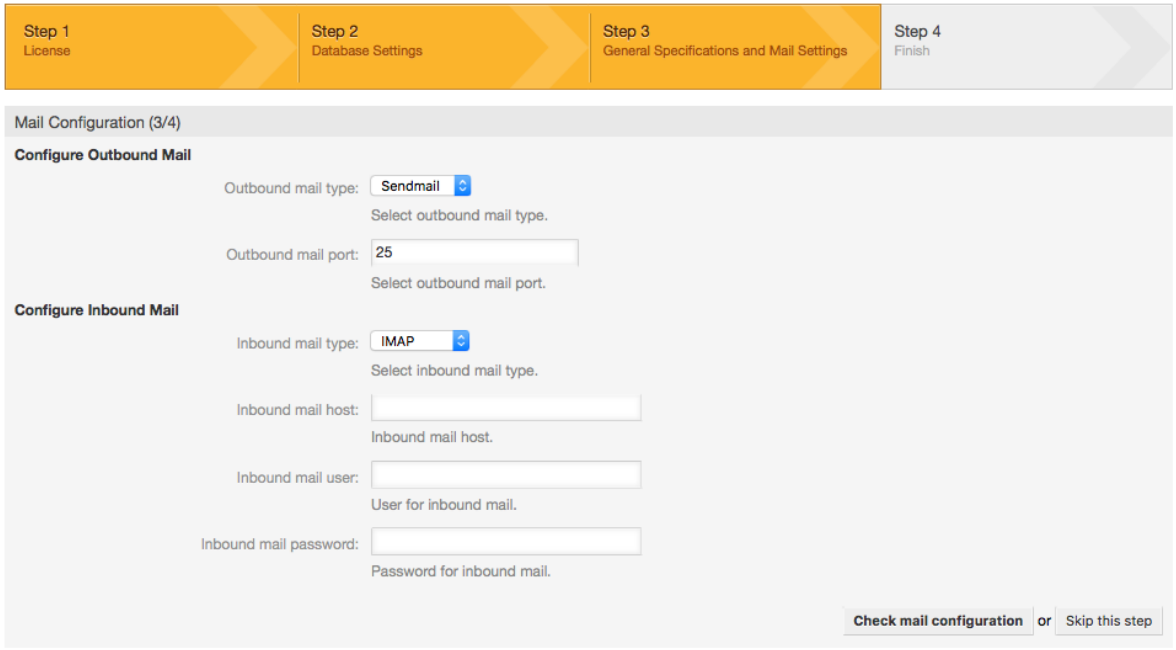
7. Provide all the required system settings and click on 'Next' (see figure below).

Figure 2.7. System settings



8. If desired, you can provide the needed data to configure your inbound and outbound mail, or skip this step by pressing the right button at the bottom of the screen (see figure below).

Figure 2.8. Mail configuration

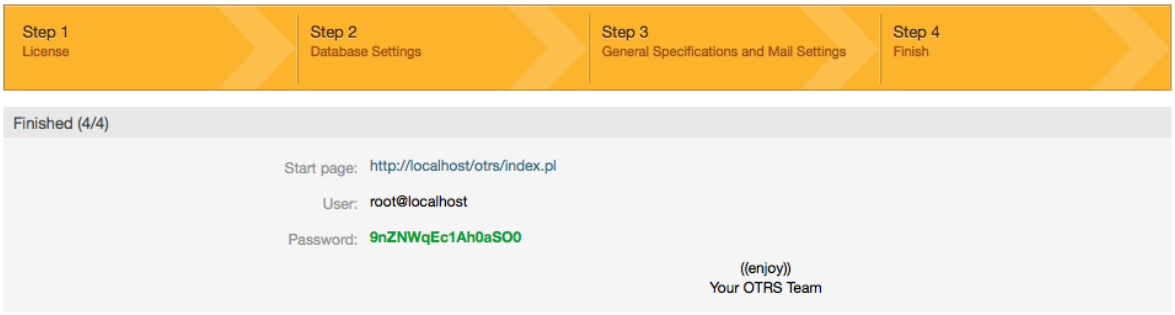


9. Congratulations! Now the installation of OTRS is finished and you should be able to work with the system (see figure below). To log into the web interface of OTRS, use the address http://localhost/otrs/index.pl from your web browser. Log in as OTRS administrator, using the username 'root@localhost' and the generated password. After that, you can configure the system to meet your needs.

Warning

Please write down the generated password for the 'root@localhost' account.

Figure 2.9. Web installer final screen



4. OTRS on Windows

OTRS can be run on a wide range of system platforms, including Enterprise Linux Platforms such as Red Hat Enterprise Linux, and SUSE Linux Enterprise Server, as well as a series of other Linux derivatives.

However, when running OTRS on Windows platforms we have encountered repeated performance losses, and despite an exhaustive analysis, it has not been possible to solve these issues to our satisfaction due to technical differences. It is thus with a heavy heart that we have ceased development on our Windows Installer and the OTRS Appliance due to the currently limited availability of necessary third-party components offered by other vendors.

Under these circumstances, we are not able to guarantee the continuing operation of OTRS on Windows platforms, and therefore recommend migrating to one of the Linux platforms mentioned above or recommend using our **OTRS Business Solution™** Managed.

To make it easier for you to migrate from Windows to Linux and to offer you the best OTRS performance, we have prepared detailed instructions for you here.

4.1. How to migrate existing Windows installations to Linux

4.1.1. Introduction and preparation

If you have a Windows based installation and you would like to change to a Linux based system you will need to setup a Linux server or virtual machine and install OTRS there (see the installation instructions). This will be the target system for the migration.

4.1.2. Get OTRSCloneDB script to clone databases

Please go to the admin menu of the Windows based system and install the newest version of OTRSCloneDB package into your OTRS:

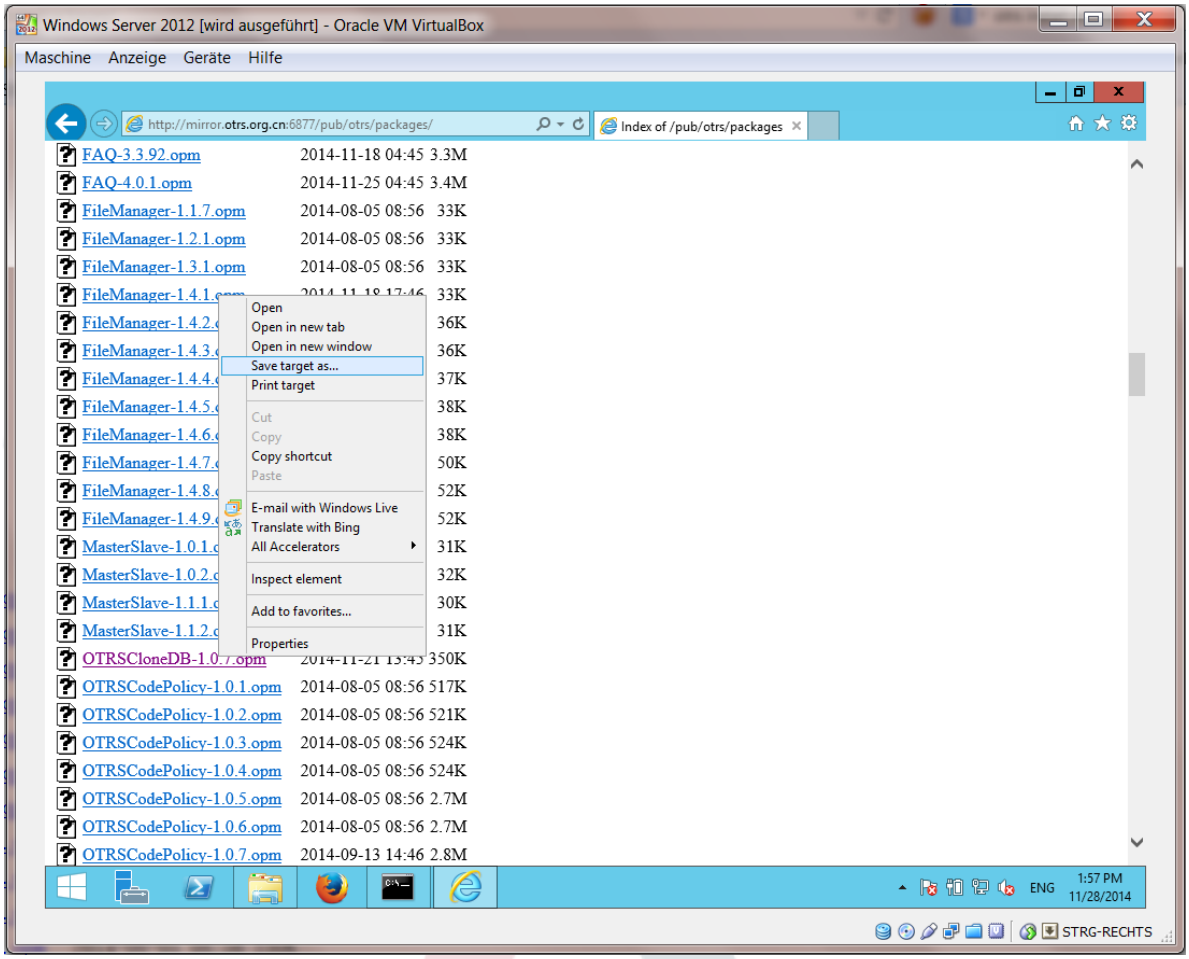
You can install the OTRSCloneDB package directly from the package manager. Select "OTRS Extensions" from the dropdown list on the left and click on the button "Update repository information" below. Then the OTRSCloneDB package will be shown in a list where you can click on "Install".

You could also download the package manually from the OTRS FTP server and install the package manually as described below. http://ftp.otrs.org/pub/otrs/packages/

Please download the package with the highest version number:

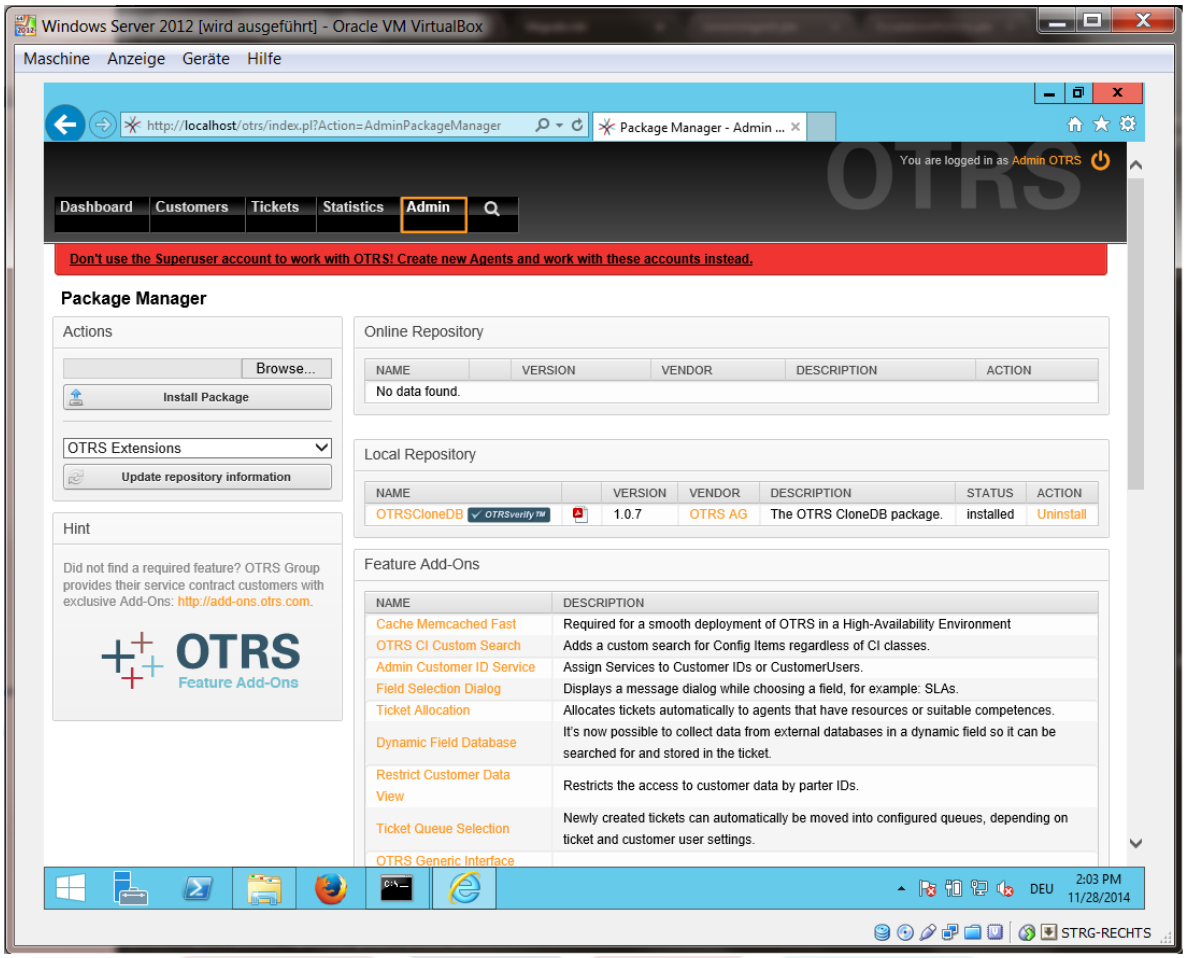
```
OTRSCloneDB-1.0.13.opm
```

Figure 2.10. Download OTRSCloneDB - screenshot



and install it to your Windows based installation:

Figure 2.11. Install OTRSCloneDB - screenshot



It is also no problem if you have installed some additional features or custom developments on your OTRS. You just need to take care that all of your installed packages are also compatible with Unix based systems. For packages provided by OTRS this is the case.

4.1.3. Enable remote access for the PostgreSQL database of target system

The OTSRCloneDB script will copy the database data over the network, so we need to enable remote access to the database. The setup is different for the different databases, we will describe opening remote access for a PostgreSQL database here.

After logging into your target system via SSH you need to change into the postgresql directory:

```
shell> cd /etc/postgresql/9.4/main
shell> vi postgresql.conf
```

Add the following line at the end of the file:

```
listen_addresses = '*'
```

Save the file.

```
shell> vi pg_hba.conf
```

Add the following line at the end of the file:

```
host all all  0.0.0.0/0 md5
```

Save the file.

Restart your PostgreSQL server:

```
shell> service postgresql restart
```

4.1.4. Stop OTRS services

Stop all running services of your target system:

```
shell> service cron stop
shell> service apache2 stop
shell> su - otrs
shell> cd /opt/otrs/
shell> bin/Cron.sh stop
shell> bin/otrs.Daemon.pl -a stop
shell> exit
```

4.1.5. Drop the existing database of your target system to have an empty database for the clone data

The OTRSCloneDB script will not remove the data in the existing otrs database of the target system, so we need to do this manually:

Change the user to the postgresql user:

```
shell> su - postgres
```

Drop the existing otrs database:

```
shell> dropdb otrs
```

Create a new otrs database for the otrs user:

```
shell> createdb --owner=otrs --encoding=utf8 otrs
```

Go back to root user:

```
shell> exit
```

4.1.6. Get the PostgreSQL password of your database

Change into the OTRS directory of your target system:

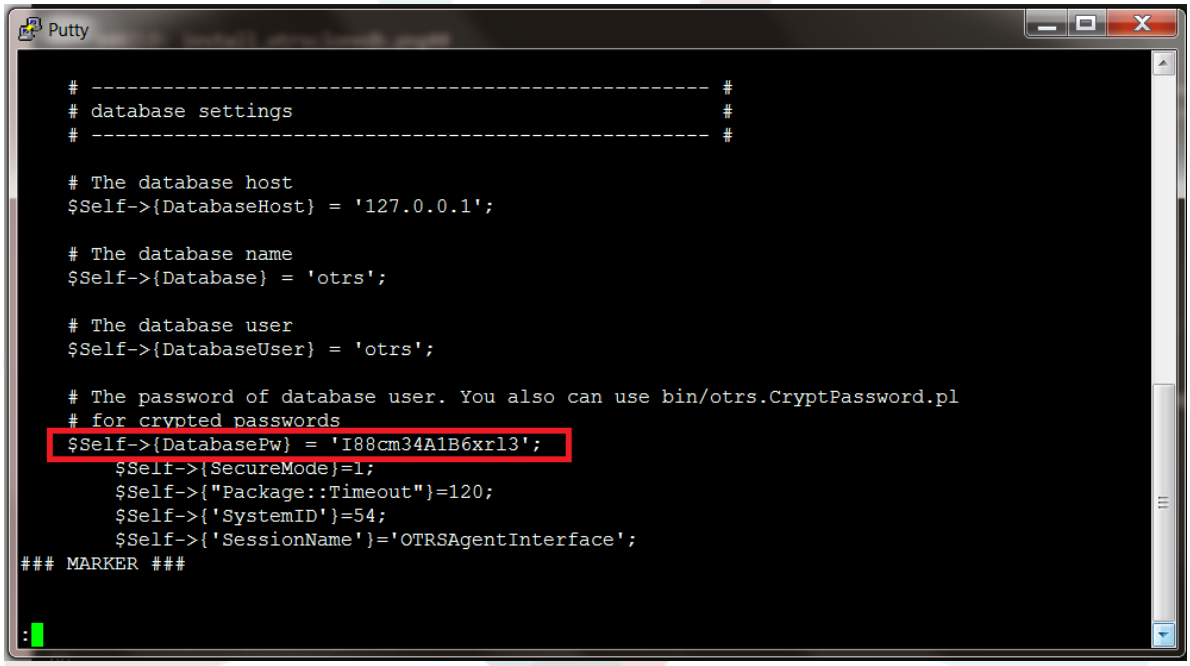
```
shell> cd /opt/otrs
```

and take a look at the configuration file of your target system:

```
shell> less Kernel/Config.pm
```

You will find your database password if you scroll down a bit:

Figure 2.12. Get target database password - screenshot



In our example:

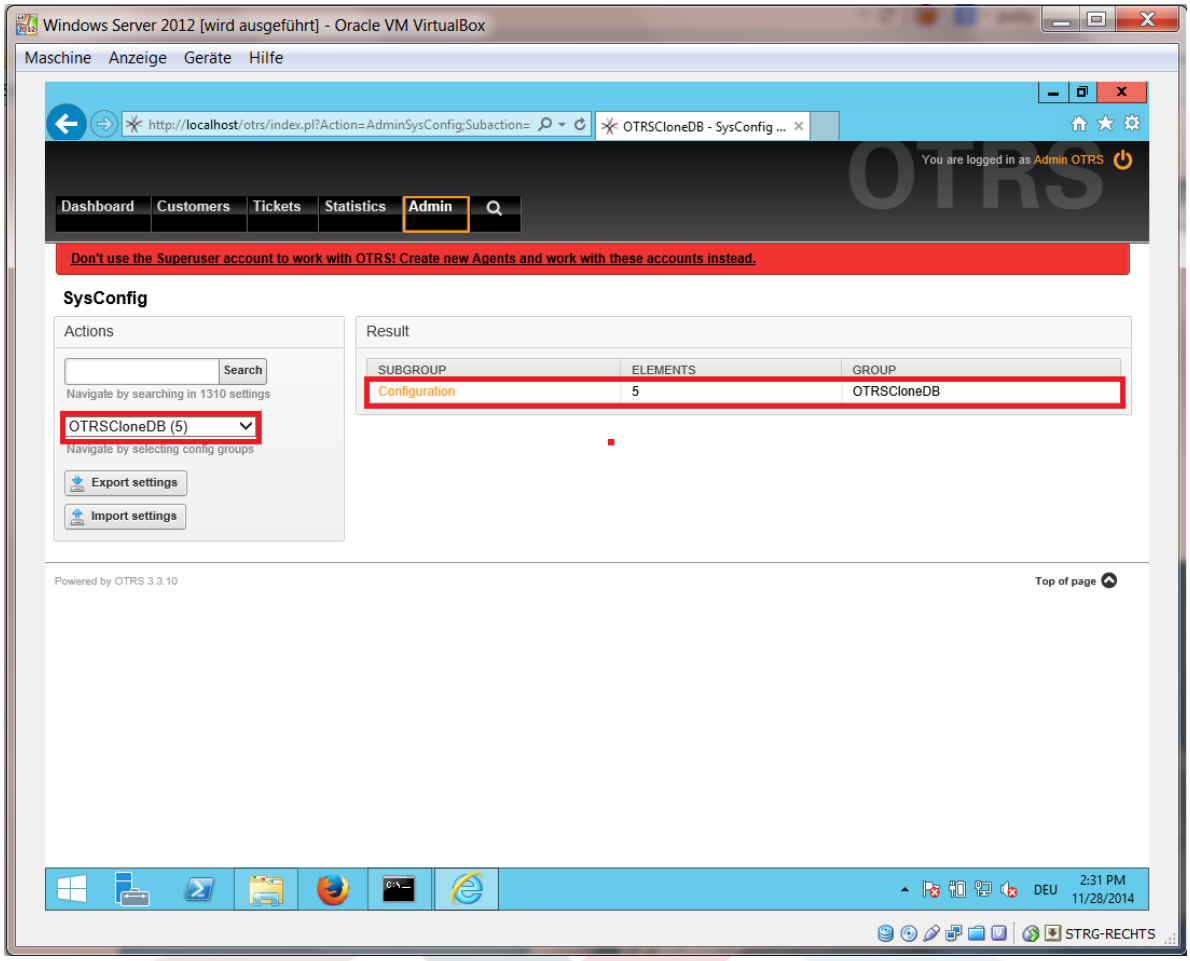
```
I88cm34A1B6xrl3
```

Write the password down on a piece of paper.

4.1.7. Clone your database into the target system

Switch back to your Windows based installation and open the SysConfig admin menu. Please select the group "OTRSCloneDB" and the subgroup "Configuration":

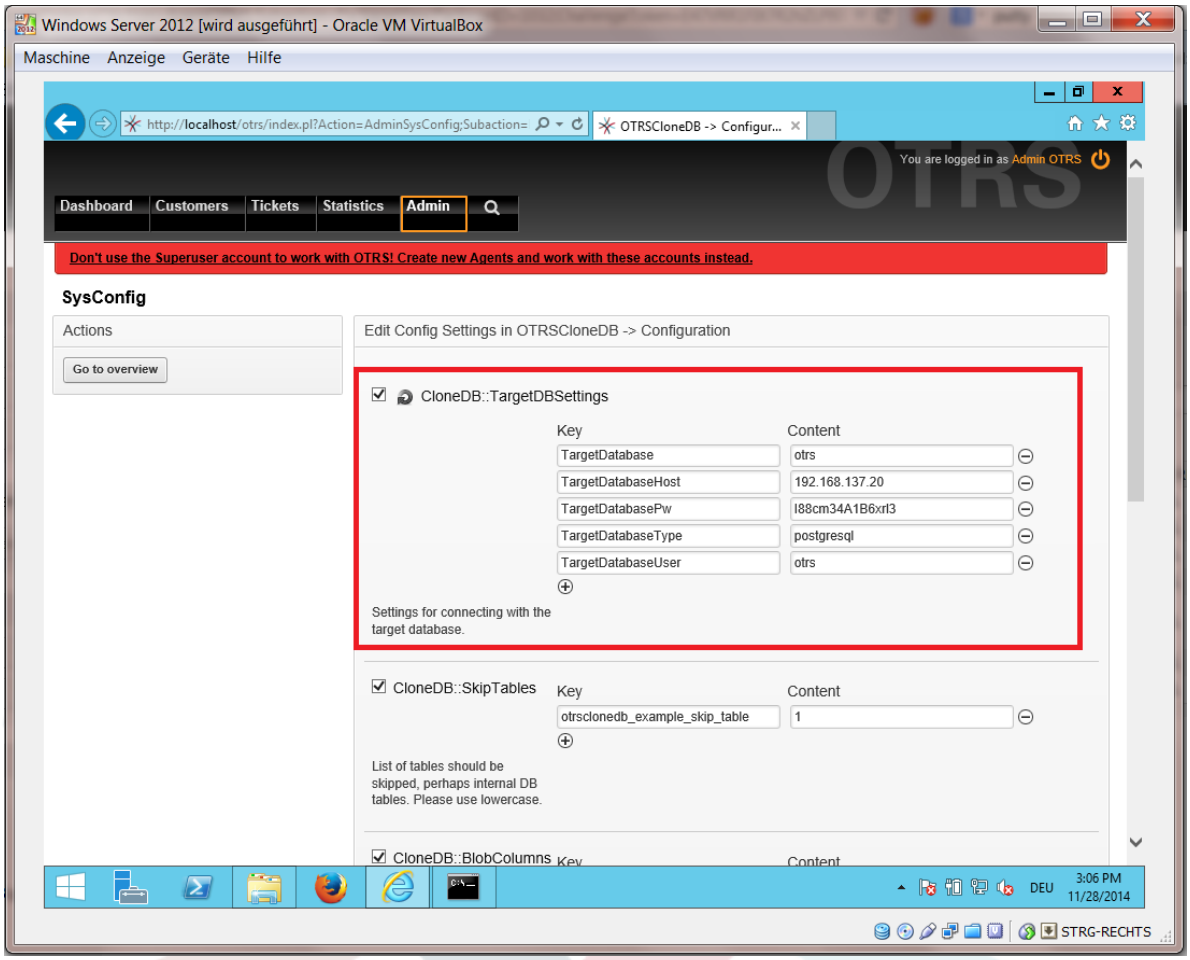
Figure 2.13. Configurate OTRSCloneDB SysConfig 1 - screenshot



We need to configure the SysConfig option OTRSCloneDB::TargetSettings with the following values:

```
TargetDatabaseHost => 192.168.137.20 (Here you need to enter the ip address of your target
 system)
TargetDatabase => otrs
TargetDatabaseUser => otrs
TargetDatabasePw => I88cm34A1B6xrl3 (Here you need to set the password of your target
 system)
TargetDatabaseType => postgresql
```

Figure 2.14. Configurate OTRSCloneDB SysConfig 2 - screenshot



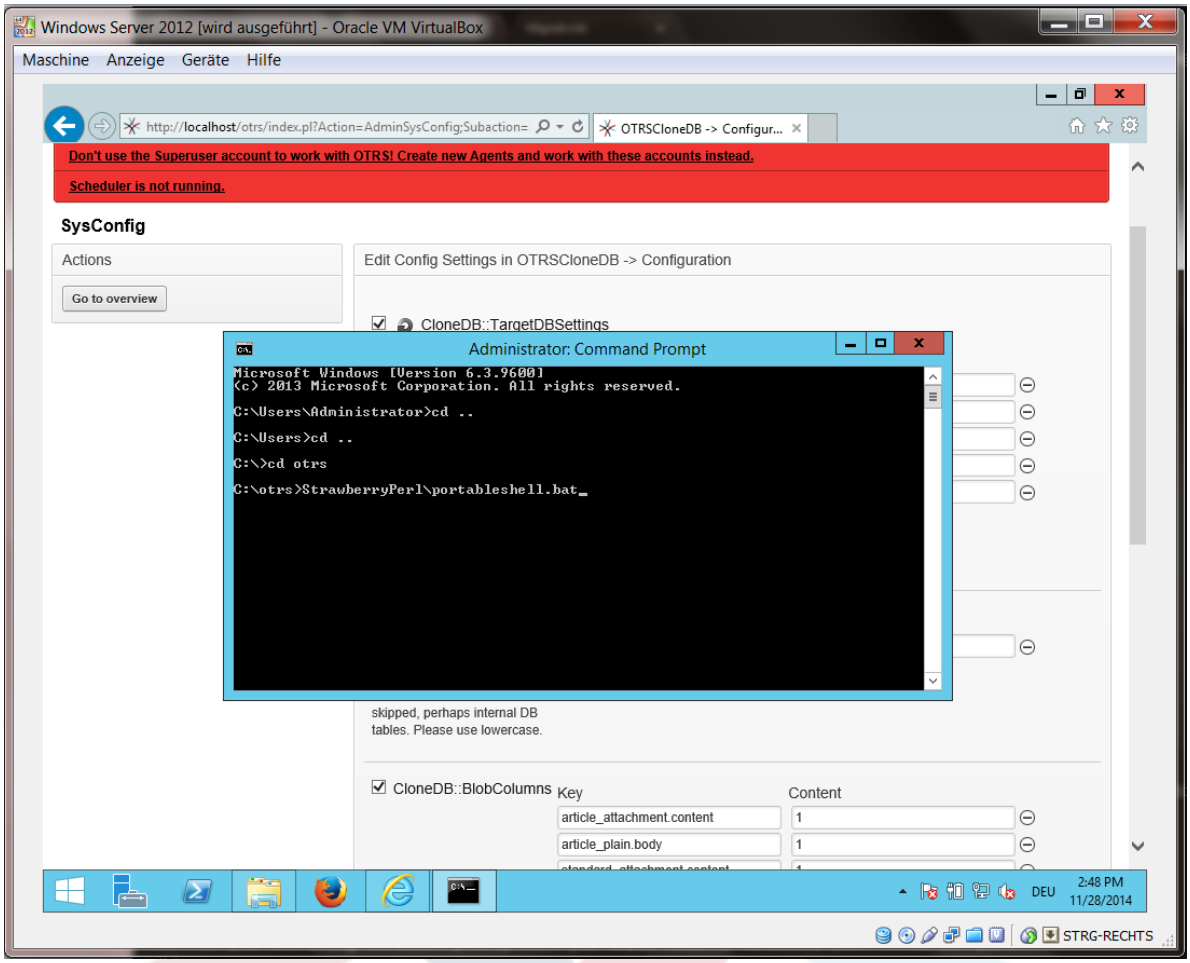
To run the OTRSCloneDB script we need to switch to the command prompt of our Windows based OTRS and to change into the base directory of our OTRS installation:

```
shell> cd "C:\otrs"
```

If you are using StrawberryPerl, then you maybe need to activate your shell for Perl:

```
shell> StrawberryPerl\portableshell.bat
```

Figure 2.15. Run OTRSCloneDB script 1 - screenshot



The OTRSCloneDB script is located in the bin directory of the OTRS directory.

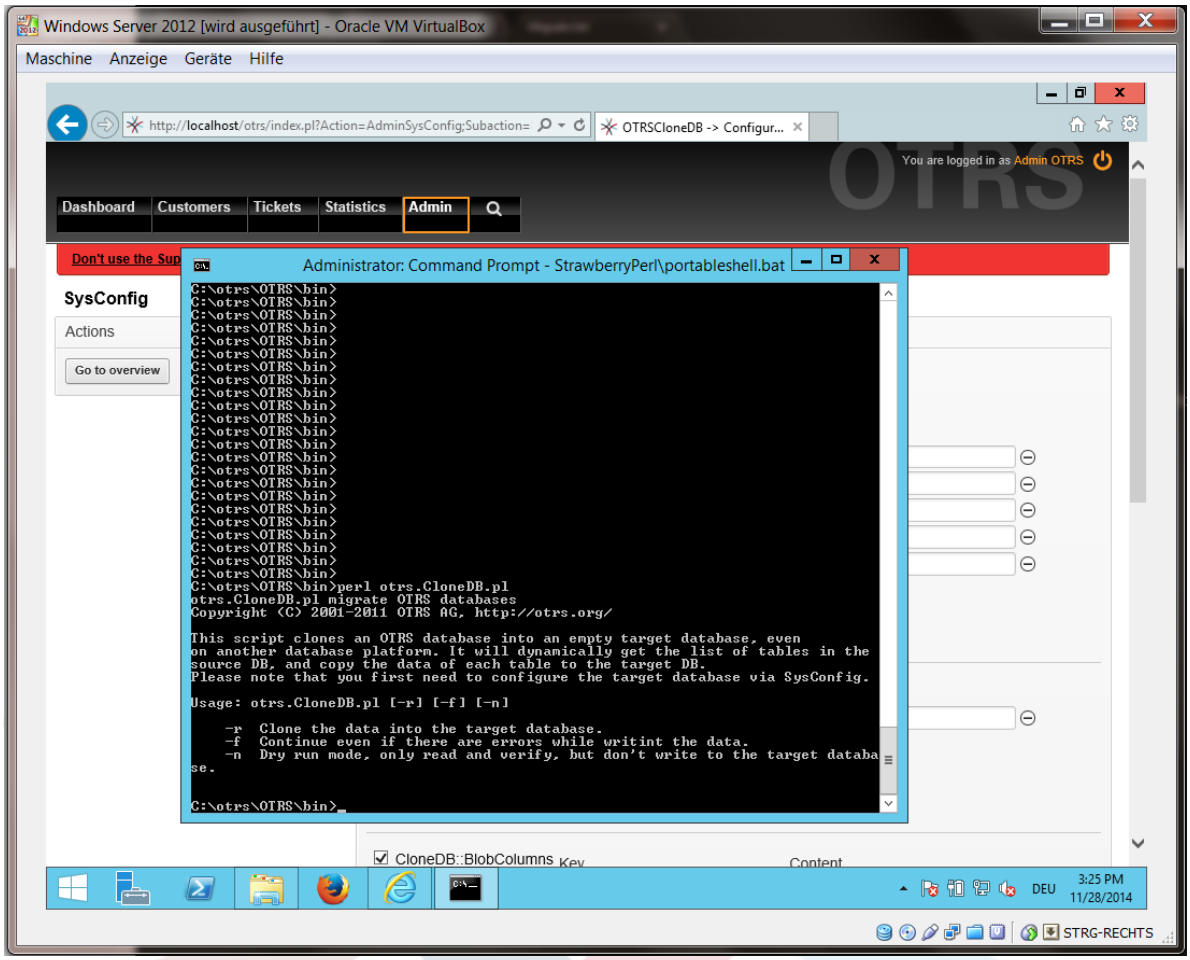
```
shell> cd "OTRS\bin"
```

Run the OTRSCloneDB script:

```
shell> perl otrs.OTRSCloneDB.pl
```

Now you should see some information about the script and its parameters.

Figure 2.16. Run OTRSCloneDB script 2 - screenshot

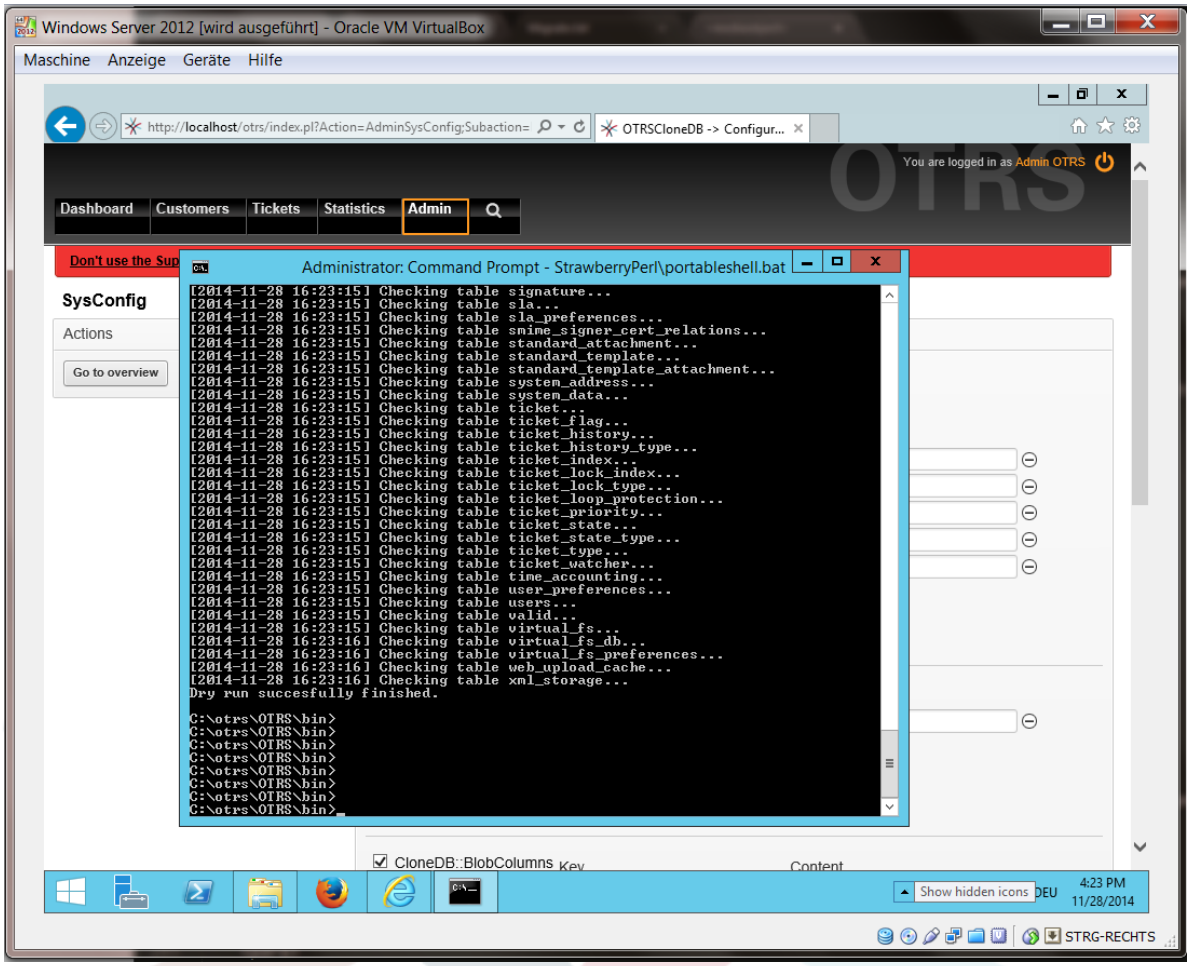


Note... Errors which occur at running without any parameters could relate to wrong login data for the database or problems with the database connection.

If no errors occurred, check if the dry run is successful:

```
shell> perl otrs.OTRSCloneDB.pl -n
```

Figure 2.17. Run OTRSCloneDB script 3 - screenshot



Start the cloning of your database and cross your fingers:

```
shell> perl otrs.OTRSCloneDB.pl -r
```

An example of a successfully run look like this:

```
Generating DDL for OTRS.
Generating DDL for package OTRSCloneDB.
Creating structures in target database (phase 1/2)
...
...
Creating structures in target database (phase 2/2)
...
done.
```

4.1.8. Copy the following files from your Windows based system to the target system

You need to copy some files from your Windows based system to the target system. You can do this for example with a free tool like "WinSCP" (just search the internet for "WinSCP"). Copy the following files from your Windows based system:

```
C:\otrs\OTRS\Kernel\Config\GenericAgent.pm
```

```
C:\otrs\OTRS\Kernel\Config\Files\ZZZAuto.pm
C:\otrs\OTRS\var\article\*
C:\otrs\OTRS\var\log\TicketCounter.log
```

to your target system:

```
/opt/otrs/Kernel/Config/GenericAgent.pm
/opt/otrs/Kernel/Config/Files/ZZZAuto.pm
/opt/otrs/var/article/*
/opt/otrs/var/log/TicketCounter.log
```

Open the file `/opt/otrs/Kernel/Config/Files/ZZZAuto.pm` on the target system and replace all paths like "C:/otrs/OTRS/" with "/opt/otrs/"!

If you have have manually changes in your `Kernel/Config.pm` then please copy these changes to the target system's `Kernel/Config.pm`. Don't copy it 1:1 because you will now have different database settings and the file paths on the target system are different from Windows!

4.1.9. Reinstall all packages

Reinstall all packages with the package manager to get all custom files back.

```
shell> bin/otrs.SetPermissions.pl --otrs-user=otrs --otrs-group=otrs --web-user=www-data --
web-group=www-data /opt/otrs
shell> su - otrs
shell> cd /opt/otrs
shell> perl bin/otrs.Console.pl Maint::Cache::Delete
shell> perl bin/otrs.Console.pl Maint::Loader::CacheCleanup
shell> perl bin/otrs.Console.pl Admin::Package::ReinstallAll
shell> exit
```

Fix all permissions in your OTRS system again:

```
shell> bin/otrs.SetPermissions.pl --otrs-user=otrs --otrs-group=otrs --web-user=www-data --
web-group=www-data /opt/otrs
```

4.1.10. Disable remote access for the PostgreSQL database of your target system

Undo all steps you did to enable the remote access for the PostgreSQL database to your target system.

Change into postgresql directory:

```
shell> cd /etc/postgresql/9.4/main
shell> vi postgresql.conf
```

Remove the following line at the end of the file:

```
listen_addresses = '*'
```

Save the file.

```
shell> vi pg_hba.conf
```

```
```

Remove the following line at the end of the file:

```
host all all  0.0.0.0/0 md5
```

Save the file.

Restart your postgresql server

```
shell> service postgresql restart
```

4.1.11. Start OTRS services

Start services of your target system:

```
shell> service cron start
shell> service apache2 start
shell> su - otrs
shell> cd /opt/otrs/
shell> bin/Cron.sh start
shell> bin/otrs.Daemon.pl -a start
```

Now you should be able to open the OTRS of your target system in the browser with the imported data of your Windows based system.

5. Upgrading OTRS from 4 to 5

These instructions are for people upgrading OTRS from *4* to *5* or from a *5* to a later patch-level release *5* and applies both for RPM and source code (tarball) upgrades.

If you are running a lower version of OTRS you have to follow the upgrade path to 4 first (1.1->1.2->1.3->2.0->2.1->2.2->2.3->2.4->3.0->3.1->3.2->3.3->4)! You need to perform a full upgrade to every version in between, including database changes and the upgrading Perl script.

Please note that if you upgrade from OTRS 2.2 or earlier, you have to take an extra step.

Within a single minor version you can skip patch level releases if you want to upgrade. For instance you can upgrade directly from OTRS 5 patchlevel 2 to version 5 patchlevel 6. If you need to do such a "patch level upgrade", you should skip steps 6, 10, 11, 14, 17 and 18.

It is highly recommended to perform a test update on a separate testing machine first.

5.1. Step 1: Stop all relevant services

Please make sure there are no more running services or cron jobs that try to access OTRS. This will depend on your service configuration, here is an example:

```
shell> /etc/init.d/cron stop
shell> /etc/init.d/postfix stop
shell> /etc/init.d/apache stop
```

Stop OTRS cron jobs and the scheduler or daemon (in this order) depending on the OTRS version you are updating from:

```
shell> cd /opt/otrs/
shell> bin/Cron.sh stop
shell> bin/otrs.Scheduler.pl -a stop
```

or

```
shell> cd /opt/otrs/
shell> bin/Cron.sh stop
shell> bin/otrs.Daemon.pl stop
```

5.2. Step 2: Backup everything below `/opt/otrs/`

- `Kernel/Config.pm`
- `Kernel/Config/GenericAgent.pm` (only for reference, this file is not needed any more)
- `Kernel/Config/Files/ZZZAuto.pm`
- `var/*`
- as well as the database

5.3. Step 3: Make sure that you have backed up everything ;-)

5.4. Step 4: Install the new release (tar or RPM)

5.4.1. Step 4.1: With the tarball:

```
shell> cd /opt
shell> mv otrs otrs-old
shell> tar -xzf otrs-x.x.x.tar.gz
shell> mv otrs-x.x.x otrs
```

5.4.1.1. Restore old configuration files

- `Kernel/Config.pm`
- `Kernel/Config/Files/ZZZAuto.pm`

5.4.1.2. Restore TicketCounter.log

In order to let OTRS continue with the correct ticket number, restore the `TicketCounter.log` to `/opt/otrs/var/log/`. This is especially important if you use incremental ticketnumbers.

5.4.1.3. Restore article data

If you configured OTRS to store article data in the filesystem you have to restore the `article` folder to `/opt/otrs/var/` or the folder specified in the SysConfig.

5.4.1.4. Set file permissions

Please execute

```
shell> cd /opt/otrs/
shell> bin/otrs.SetPermissions.pl
```

with the permissions needed for your system setup. For example:

- Web server which runs as the OTRS user:

```
shell> bin/otrs.SetPermissions.pl --web-group=otrs
```

- Webserver with wwwrun user (e. g. SUSE):

```
shell> bin/otrs.SetPermissions.pl --web-group=wwwrun
```

- Webserver with apache user (e. g. Red Hat, CentOS):

```
shell> bin/otrs.SetPermissions.pl --web-group=apache
```

- Webserver with www-data user (e. g. Debian, Ubuntu):

```
shell> bin/otrs.SetPermissions.pl --web-group=www-data
```

5.4.2. Step 4.2: With the RPM:

```
shell> rpm -Uvh otrs-x.x.x.-01.rpm
```

In this case the RPM update automatically restores the old configuration files and sets file permissions.

5.5. Step 5: Check needed Perl modules

Verify that all needed Perl modules are installed on your system and install any modules that might be missing.

```
shell> /opt/otrs/bin/otrs.CheckModules.pl
```

5.6. Step 6: Apply the database changes

5.6.1. Step 6.1: Database schema update

5.6.1.1. MySQL:

Note

Note: new tables created in the MySQL UPGRADING process will be created with the default table storage engine set in your MySQL server. In MySQL 5.5 the new default type is InnoDB. If existing tables, e.g. "users", have the table storage en-

gine e.g. MyISAM, then an error will be displayed when creating the foreign key constraints. In this case we recommend to switch all tables to InnoDB with the console command **bin/otrs.Console.pl Maint::Database::MySQL::InnoDBMigration**.

Any problems with regards to the storage engine will be reported by the bin/otrs.Console.pl Maint::Database::Check command, so please run it to check for possible issues.

```
shell> cd /opt/otrs/
shell> cat scripts/DBUpdate-to-5.mysql.sql | mysql -p -f -u root otrs
shell> bin/otrs.Console.pl Maint::Database::Check
```

5.6.1.2. PostgreSQL:

```
shell> cd /opt/otrs/
shell> cat scripts/DBUpdate-to-5.postgresql.sql | psql --set ON_ERROR_STOP=on --single-
transaction otrs otrs
```

5.6.2. Step 6.2: Database migration script

Run the migration script (as user otrs, NOT as root):

```
shell> scripts/DBUpdate-to-5.pl
```

Do not continue the upgrading process if this script did not work properly for you. Otherwise data loss may occur.

5.7. Step 7: Refresh the configuration cache and delete caches

Please run (as user otrs, *not* as root):

```
shell> cd /opt/otrs/
shell> bin/otrs.Console.pl Maint::Config::Rebuild
shell> bin/otrs.Console.pl Maint::Cache::Delete
```

5.8. Step 8: Restart your services

e. g. (depends on used services):

```
shell> /etc/init.d/apache start
shell> /etc/init.d/postfix start
shell> /etc/init.d/cron start
```

Now you can log into your system.

5.9. Step 9: Check installed packages

Note

The OTRS packages of 4 are NOT compatible with OTRS 5, so you have to perform a package upgrade!

The following packages are automatically uninstalled after the upgrade process (if they where installed before):

- OTRSGenericInterfaceMappingXSLT

5.10. Step 10: Configure NodeIDs (only for multi-frontend clustered setups)

Note

This step is only needed if you have a clustered setup with several frontend machines.

From OTRS 5 on, every frontend server needs to have its own unique `NodeID`. This defaults to 1 and thus does not need to be configured for single-frontend setups. If you have more than one machine, each machine needs to have this value set to a unique value between 1 and 999. This configuration needs to be done in the file `Kernel/Config.pm`:

```
$Self->{'NodeID'} =  '2';   # assign a unique value for every frontend server
```

5.11. Step 11: Check follow-up detection configuration

The follow-up detection settings were reorganized. Now OTRS searches by default in email subject and references to detect follow-ups. Please check in AdminSysConfig `Ticket -> Core::PostMaster` if you need to make any changes to the follow-up detection configuration (for example to search in body, attachments or raw email content).

5.12. Step 12: Start the OTRS Daemon

The new OTRS daemon is responsible for handling any asynchronous and recurring tasks in OTRS. What has been in cron file definitions previously is now handled by the OTRS daemon, which is now required to operate OTRS. The daemon also handles all GenericAgent jobs and must be started from the `otrs` user.

```
shell> /opt/otrs/bin/otrs.Daemon.pl start
```

5.13. Step 13: Update and activate cron jobs

There are two default OTRS cron files in `/opt/otrs/var/cron/*.dist`, and their purpose is to make sure that the OTRS Daemon is running. They need to be activated by copying them without the ".dist" filename extension.

```
shell> cd /opt/otrs/var/cron
shell> for foo in *.dist; do cp $foo `basename $foo .dist`; done
```

To schedule these cron jobs on your system, you can use the script `Cron.sh` with the `otrs` user.

```
shell> /opt/otrs/bin/Cron.sh start
```

Please note that if you had any custom cron jobs, you should consider moving them to SysConfig (`Daemon -> Daemon::SchedulerCronTaskManager::Task`) to have them executed by the OTRS daemon as well. You might also need to adapt your custom scripts, because now most OTRS commands are managed by `bin/otrs.Console.pl` instead of single scripts.

Note

The console command `Dev::Code::Generate::ConsoleCommand` can help to create new console commands for your custom scripts. This command creates a template where the script logic and its parameters can be adapted.

5.14. Step 14: Review your ticket notifications

With OTRS 5 ticket notifications are configured differently than in previous versions. They are now part of the "Event Notifications" that were previously available as well (now called just "Ticket Notifications"). Your existing ticket notifications have been migrated into the list of ticket notifications, but deactivated. You have also received the new default ticket notifications, active by default.

In case you did not modify the old ticket notifications you don't have to do anything. If you have made local modifications or translations, you have two choices: a) you can migrate your modifications to the new ticket notifications (recommended) or b) you can deactivate the new notifications and activate the old notifications again.

Please also review the escalation notification recipient settings and see if they match with your expectations (agents who have the ticket in one of their custom queues vs. all agents with read permissions). This was configured via a setting in the file `Kernel/Config/GenericAgent.pm` (now obsolete) previously and can now be controlled via the GUI for each notification separately.

5.15. Step 15: Update system registration (optional)

If the system is already registered with OTRS Group, it is strongly recommended to update the registration information at this time. This will update the registered version of the system (among other changes) in the OTRS Group records, in order to get much accurate information from the cloud services.

If you don't update the registration information manually, it will be done automatically on a regular basis, but this could happen some hours or days after. During this period it might be possible to get wrong information from cloud services like **OTRS Business Solution™** updates.

```
shell> cd /opt/otrs/
shell> bin/otrs.Console.pl Maint::Registration::UpdateSend --force
shell> bin/otrs.Console.pl Maint::Cache::Delete
```

5.16. Step 16: Migrate custom Perl based GenericAgent jobs (optional)

This is only relevant if you had any custom GenericAgent jobs in Perl OTRS 4 configuration files such as `Kernel/Config/GenericAgent.pm` that cannot be handled with the regular

ticket notifications. Such jobs now need to be registered as OTRS Daemon cron tasks in the SysConfig (Daemon -> Daemon::SchedulerCronTaskManager::Task), in order to be executed on a regular basis.

There are 5 settings in the SysConfig prepared for this purpose (Daemon::SchedulerCronTaskManager::Task###GenericAgentFile1 to Daemon::SchedulerCronTaskManager::Task###GenericAgentFile5). If more settings are needed they can be added in a custom SysConfig setting file.

Please replace "<ModuleName>" with the file that contains the custom GenericAgent jobs configuration, e.g. Kernel::Config::GenericAgent for the file: Kernel/Config/GenericAgent.pm, set the schedule to match the periodicity as it was executed before, mark the setting as active and save the changes.

Note

To get more information about other parameters that can be used while running the GenericAgent jobs, please execute:

```
shell> bin/otrs.Console.pl Maint::GenericAgent::Run --help
```

5.17. Step 17: Setup bash autocompletion (optional)

All regular OTRS command line operations now happen via the OTRS Console interface bin/otrs.Console.pl. This provides an auto completion for the bash shell which makes finding the right command and options much easier.

You can activate the bash autocompletion by installing the package bash-completion. It will automatically detect and load the file /opt/otrs/.bash_completion for the otrs user.

After restarting your shell, you can just type bin/otrs.Console.pl followed by TAB, and it will list all available commands. If you type a few characters of the command name, TAB will show all matching commands. After typing a complete command, all possible options and arguments will be shown by pressing TAB.

5.18. Step 18: Review ticket action screen configurations (optional)

Some ticket action screens like "Note" had default subjects in OTRS 4 (configurable via SysConfig, Ticket::Frontend::AgentTicketNote###Subject in this case). These default subjects have been removed to reduce the amount of potentially redundant/meaningless data. You might want to re-add them if they are mandatory for you.

5.19. Step 19: Well done!

6. Additional Applications

You can install additional application packages to extend the functionality of the OTRS framework. This can be done via the package manager from the Admin page, which downloads the applications from an online repository and manages package dependencies. It is also possible to install packages from local files.

6.1. FAQ

The FAQ is the Knowledge Base component. It supports editing and viewing of FAQ articles. Articles can be viewed and restricted to agents, customer users, or anonymous users. These can also be structured into groups, and be read in different languages.

Chapter 3. First Steps

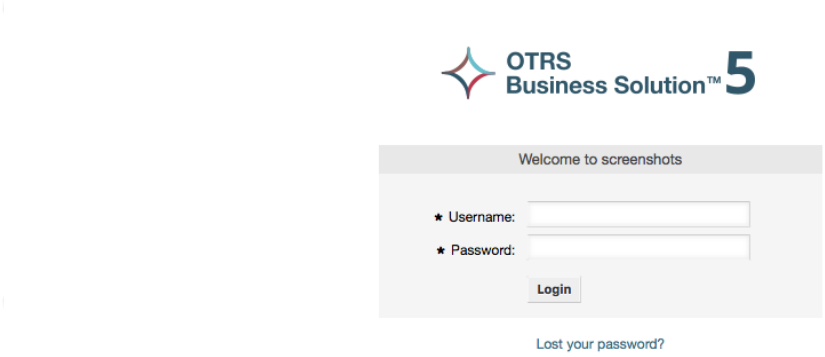
The goal of this chapter is to provide a brief overview of OTRS and the structure of its web interface. The terms 'agents', 'customers', and 'administrators' are introduced. We also login as the OTRS admininstrator and take a closer look at the user preferences available on every account.

1. Agent Web Interface

The agent web interface allows agents to answer customer requests, create new tickets for customers or other agents, write tickets about telephone calls with customers, write FAQ entries, edit customer data, etc.

Supposing your OTRS host is reachable via the URL *http://www.example.com* , then the OTRS login screen can be reached by using the address http://www.example.com/otrs/index.pl in a web browser (see figure below).

Figure 3.1. Login screen of the agent interface

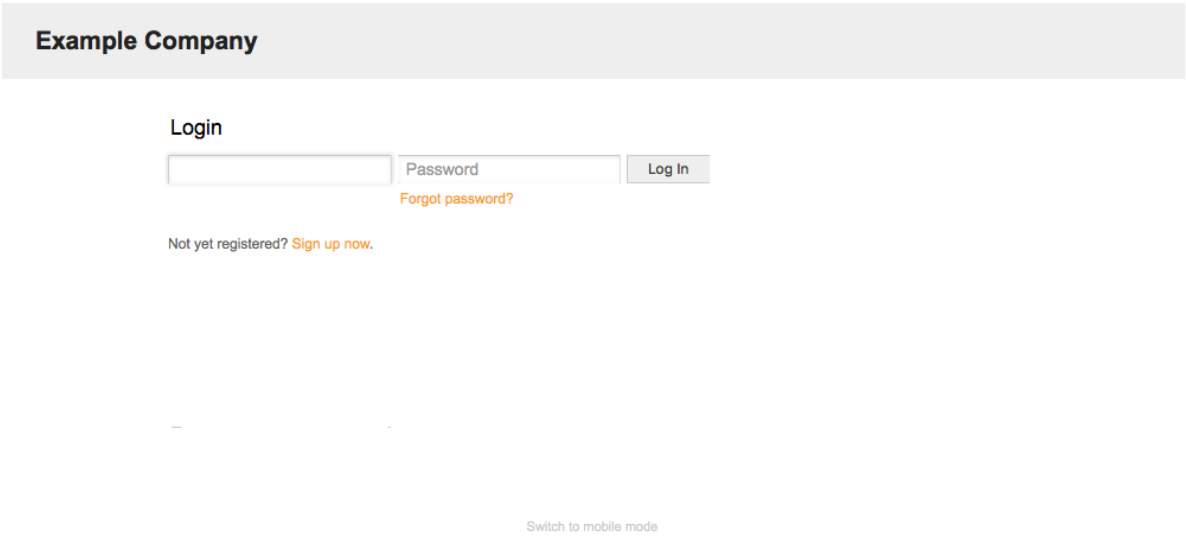


2. Customer Web Interface

Customers have a separate web interface in OTRS through which they can create new accounts, change their account settings, create and edit tickets, get an overview on tickets that they have created, etc.

Continuing the above example, the customer login screen can be reached by using the URL http://www.example.com/otrs/customer.pl with a web browser (see figure below).

Figure 3.2. Login screen of the customer interface

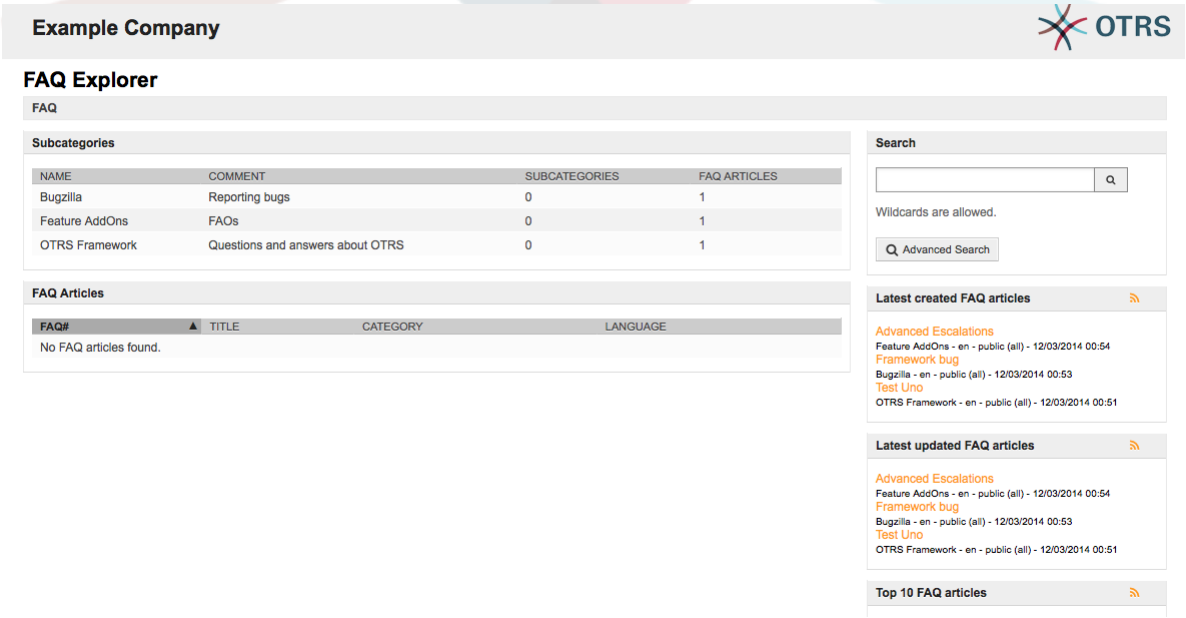


3. Public Web Interface

In addition to the web interfaces for agents and customers, OTRS also has a public web interface which is available through the FAQ-Module. This module needs to be installed separately. It provides public access to the FAQ system and lets visitors search through FAQ entries without any special authorization.

In our example, the public web interface can be reached via either of the following URLs: http://www.example.com/otrs/faq.pl , http://www.example.com/otrs/public.pl

Figure 3.3. Public web interface



4. First Login

Access the login screen as described in the section Agent web interface . Enter a user name and password. Since the system has just been installed and no users have yet been

created, login as OTRS administrator first, using 'root@localhost' for username and 'root' for password.

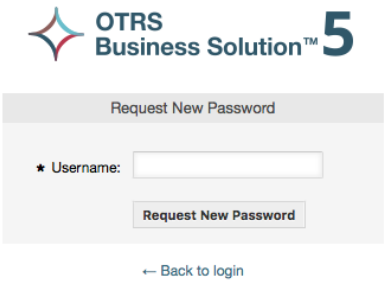
Warning

This account data is valid on every newly installed OTRS system. You should change the password for the OTRS administrator as soon as possible! This can be done via the preferences screen for the OTRS administrator account.

If you don't want to login as OTRS administrator, just enter the user name and password for your normal agent account.

In case you have forgotten your password, you can request the system for a new password. Simply press the link below the Login button, enter the mail address that is registered for your OTRS account into the input field, and press the Submit button (see figure).

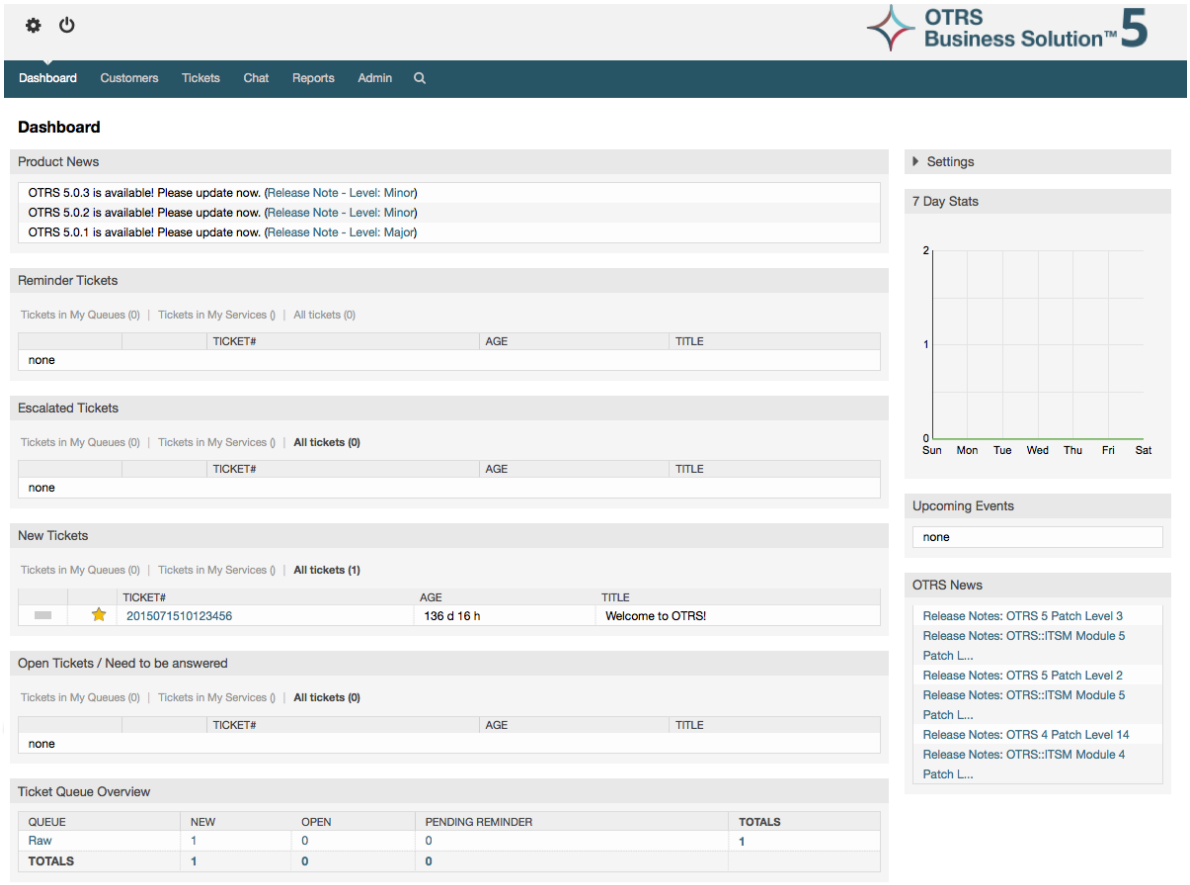
Figure 3.4. Request new password



5. The Web Interface - an Overview

Upon successfully logging into the system, you are presented with the Dashboard page (see figure below). It shows your locked tickets, allows direct access through menus to the queue, status and escalation views, and also holds options for creation of new phone and e-mail tickets. It also presents a quick summary of the tickets using different criteria.

Figure 3.5. Dashboard of the agent interface



To improve clarity, the general web interface is separated into different areas. The top row of each page shows some general information such as the logout button, icons listing the number of locked tickets with direct access to them, links to create a new phone/e-mail ticket, etc. There are also icons to go to the queue, status, and escalation views.

Below the icons row is the navigation bar. It shows a menu that enables you to navigate to different areas or modules of the system, letting you execute some global actions. Clicking on the Dashboard button takes you to the dashboard. If you click on the Tickets button, you will get a submenu with options to change the ticket's view, create a new ticket (phone/e-mail) or search for a specific ticket. The Statistics button presents a menu that allows you to choose from an overview of the registered statistics, creating a new one or importing an existing one. The Customers button leads you to the Customer Management screen. By clicking the Admin button, you can access all of the administrator modules, which allows you to create new agents, queues, etc. There is also a Search button to make ticket searches.

If any associated applications are also installed, e.g. the FAQ or the Survey, buttons to reach these applications are also displayed.

In the area below the navigation bar, different system messages can be shown. If you are logged in as the default OTRS administrator user, you get a red message warning you not to work using this system account.

Below the title of the section you are currently in, there are several subsections containing relevant information about the screen you are working on, each one in a separate box.

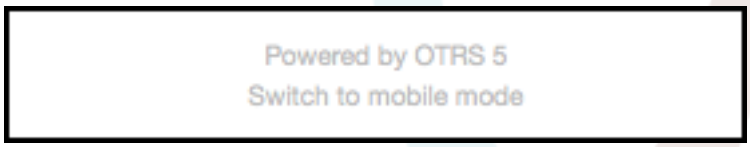
These boxes contains the main part of each screen, usually they are displayed in one or several columns, each box can store relevant information about the current screen like for

example instructions, advises, overviews, etc. Also is displayed the form or tool necessary for performing the action associated to each screen, like for example, add, update or delete records, check the log, change configuration settings, etc.

Finally at the bottom of the page, the site footer is displayed (see figure below). It contains links to directly access the OTRS official web site, or go to the Top of the page.

Normally the icon row, navigation bar and footer are the same in all the screens over the web interface.

Figure 3.6. Footer



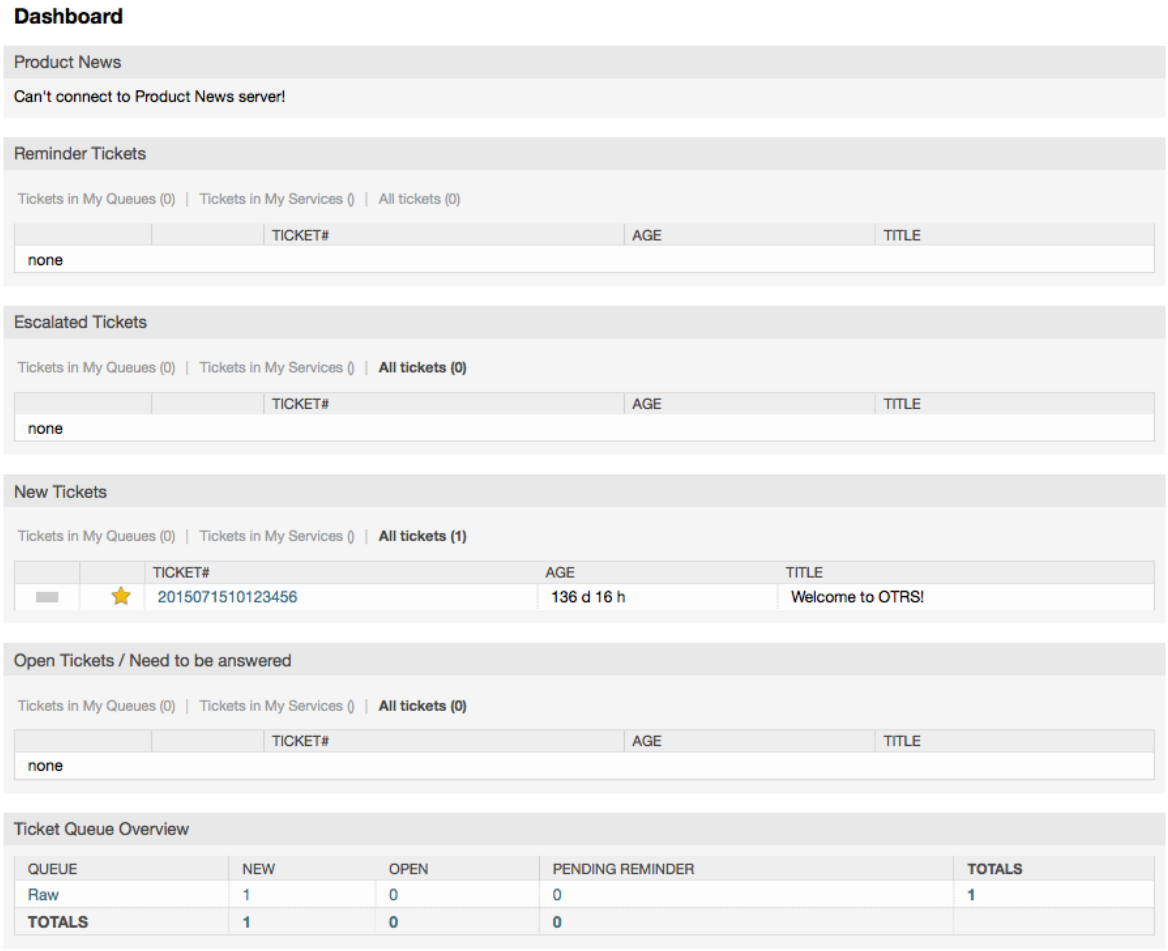
6. The Dashboard

The Dashboard is the main page of the system, here you can get an overview about the tickets and other stuff related to the ticket activity. It's thinking to be the starting point for the daily work of an agent, by default it presents a quick summary of the tickets which are pending, escalated, new, and open, among other information.

One of the most important features about Dashboard is that is completely customizable. That means you that can configure each part as you want, showing or hiding elements. It's even possible to relocate this elements within the same column by clicking on and dragging the element's header, and dropping them elsewhere. Each element is named "Widget", the system has some widgets ready to use out of the box, but the modular design of the dashboard screen is prepared to integrate custom widgets easily.

The content of this screen is arranged in two main columns, on the left column you normally can see information about tickets classified by their states like: reminder, escalated, new, and open. On each of this widgets you can filter the results to see all of the tickets that you are allowed to access, tickets you have locked, the ones that are located in agent defiend queues, among other filters. There are also other kind of widgets in this column and they are all described below.

Figure 3.7. Dashboard widgets



Left column dashboard widgets.

- Ticket List Widgets

  Widgets under this category share same overall behavior, look and feel. This widgets shows a list of tickets on a determined state. The amount of tickets display on each list page can be configured in widget options (they appear when you hover the mouse pointer over the top right part of the widget). This widgets support the following filters:

  - My locked tickets

    The tickets that the logged agent has locked.

  - My watched tickets

    The tickets that the logged agent has in his/her watched list, requires Ticket::Watcher setting to be turned on to be displayed.

  - My responsibilities

    The tickets that the logged agent is set as responsible, Ticket::Responsible setting is required to be turned on in order to make this filter visible.

  - Tickets in My Queues

    The tickets that are on queues where the agent define as "My Queues".

- Tickets in My Services

  The tickets that are assigned to services where the agent define as "My Services" and are on queues with at least read-only permissions.

- All Tickets

  All the tickets where the agent has access.

This widgets are:

- Reminder Tickets

  Tickets that are set as pending and the reminder date has been reach.

- Escalated Tickets

  Tickets that are escalated.

- New Tickets

  Tickets that have state "New".

- Open Tickets / Need to be answered

  Tickets that have state "Open" and are ready for work with them.

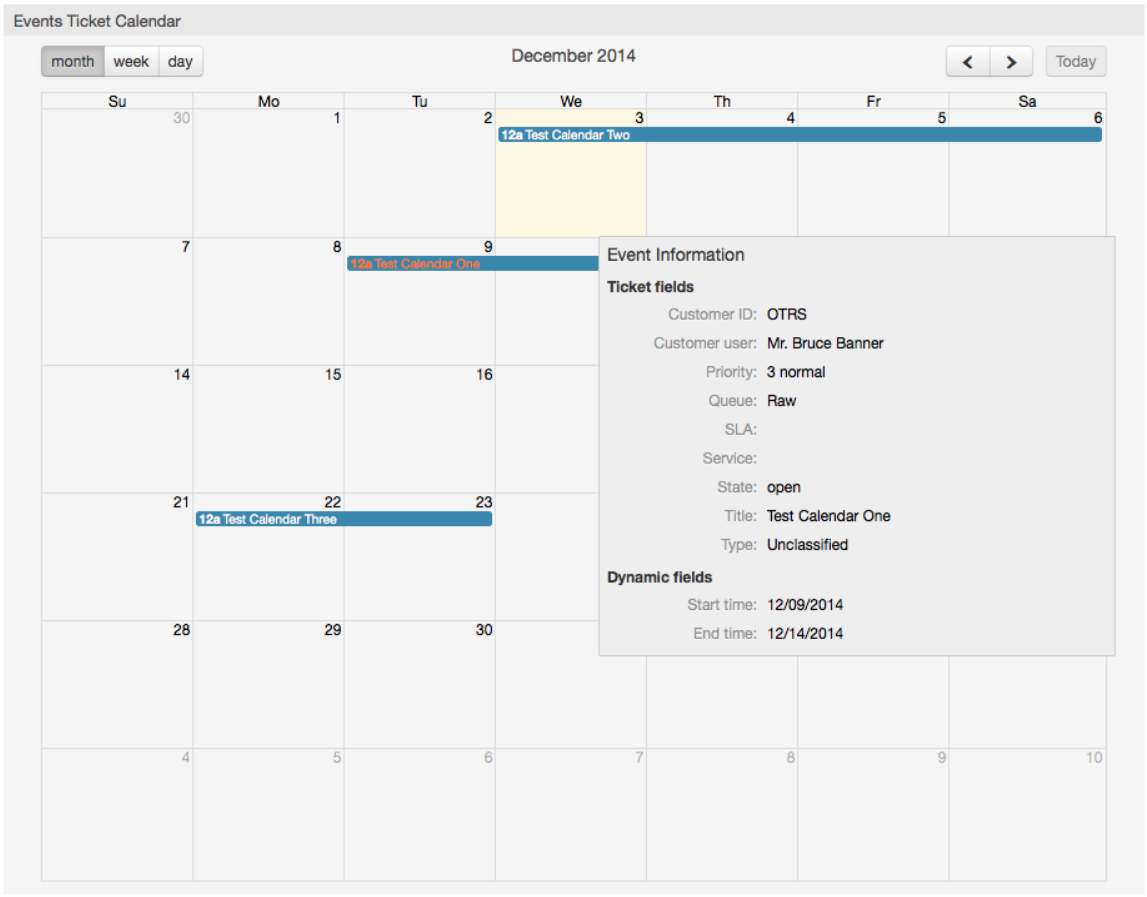
- Events Ticket Calendar

A calendar event (for this widget) is defined when a new ticket is created, the Events Ticket Calendar feature has to be enabled, and it requires two new fields to be displayed in ticket creation screens, one for the event start time and the other one for the end time, this times determine the duration of the event.

This widget includes the following views: month, week and day. Agents can scroll through the pages by using the right and left arrows.

As mentioned before just enabling the widget is not enough, a couple of "Date/Time" dynamic fields for tickets should be added into the system (via Dynamic Fields link in "Admin" panel) and set them up in the SysConfig for this widget, both Dynamic Fields should be configured to be displayed on the ticket creation screens, they should be filled during ticket creation or any other ticket action screen (e.g. Free Fields) to describe the time frame for the calendar event (start and end time), the ticket zoom screen might be configured to show this dynamic fields also, in case you consider it necessary.

Figure 3.8. Events Ticket Calendar widget



Further configurations for this widget could be found under the "Frontend::Agent::Dashboard::EventsTicketCalendar" SubGroup in the SysConfig:

- CalendarWidth

  Defines the calendar width in percent. Default is 95%.

- DynamicFieldStartTime

  Defines dynamic field name for start time.

- DynamicFieldEndTime

  Defines dynamic field name for end time.

- Queues

  Only the tickets on the queues specified in this setting will be considered in the calendar view.

- DynamicFieldsForEvents

  Defines the dynamic fields that will be displayed in the calendar event overlay windows.

- TicketFieldsForEvents

Defines the ticket attributes that will be displayed in the calendar event overlay windows.

- Ticket Queue Overview

  This widget shows in a ticket count matrix where the rows represents queues and the columns represents the ticket states, then on each cell the number of tickets on a defined state that belongs on a particular queue is displayed.

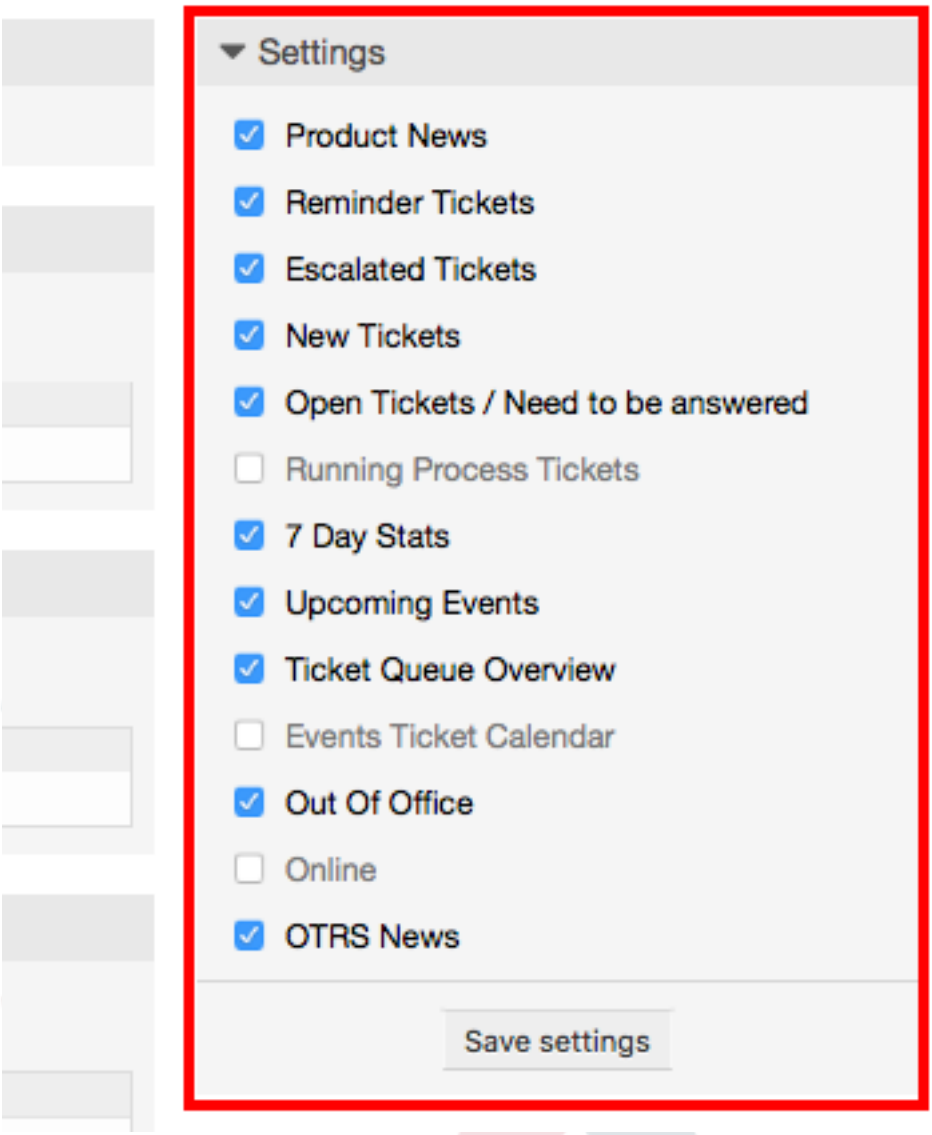
  The widget also shows a Totals row and a Totals column, the Totals row shows the sum of the tickets for each state on all presented queues, while the Totals column represent the sum of the tickets for each queue on all presented states.

  The queues and states that are presented can be changed via Sysconfig.

  By clicking any of the ticket count numbers a ticket search results page will opened letting agents to have a more detailed overview of them.

In the right column is located a special widget that allow you to control the widgets you want to show or hide. This is the Settings widget. Click on it's header to expand the section and see all available widgets, as shown in Figure. Each widget name has a checkbox. Use this checkboxes to define the visibility of the widgets in the dashboard (unchecked widgets will not be shown) after you define the visibility options and click on 'Save' for storing your changes. This section is fixed on the screen, this means you can't drag and drop it, or close it.

Figure 3.9. Dashboard Settings



Right column dashboard widgets.

- 7 Day Stats

  It shows a graph of ticket activity over the past 7 days that includes 2 lines. One that is usually blue color, represents the amount of created tickets per day and the second one, usually orange and represents the closed tickets per day.

- Upcoming Events

  Tickets on short for escalating or already escalated are listed here, info from this widget is very helpful since you have the chance to know about tickets needs your attention and you can decide in which ones you want to focus your effort on, set priorities or simply check what's coming on.

- OTRS News

  A complete list about OTRS activities and so important information about new product releases or patches.

- Online

Here is showed a summary about the current agents logged the system, it also includes a section for customers on-line, please notice this widget is normally hidden, it can be shown using Settings widget described before.

7. What is a Queue?

On many mail systems, it is common for all messages to flow into an Inbox file, where they remain stored. New messages are appended at the end of the Inbox file. The mail client program used to read and write mails reads this Inbox file and presents the content to the user.

A queue in OTRS is somewhat comparable to an Inbox file, since it too can store many messages. A queue also has features beyond those of an Inbox mail file. As an OTRS agent or user, one needs to remember which queue a ticket is stored in. Agents can open and edit tickets in a queue, and also move tickets from one queue to another. But why would they move tickets?

To explain it more practically, remember the example of Max's company described in an example of a ticket system. Max installed OTRS in order to allow his team to better manage support for company customers buying video recorders.

One queue holding all requests is enough for this situation. However, after some time Max decides to also sell DVD recorders. Now, the customers have questions not only about the video recorder, but also about the new product. More and more emails get into the single queue of Max's OTRS and it's difficult to have a clear picture of what's happening.

Max decides to restructure his support system, and adds two new queues. So now three queues are being used. New messages arriving at the ticket system are stored into the old queue titled "raw". Of the two new queues, one titled "video recorder" is exclusively for video recorder requests, while the other one titled "dvd recorder" is exclusively for dvd recorder requests.

Max asks Sandra to watch the "raw" queue and sort (dispatch) the messages either into "video recorder" or "dvd recorder" queue, depending on the customer request. John only has access to the "video recorder" queue, while Joe can only answer tickets in the "dvd recorder" queue. Max is able to edit tickets in all queues.

OTRS supports access management for users, groups, and roles, and it is easy to setup queues that are accessible only to some user accounts. Max could also use another way to get his requests into the different queues, with filter rules. Otherwise, if two different mail addresses are used, Sandra only has to dispatch those emails into the two other queues, which can't be dispatched automatically.

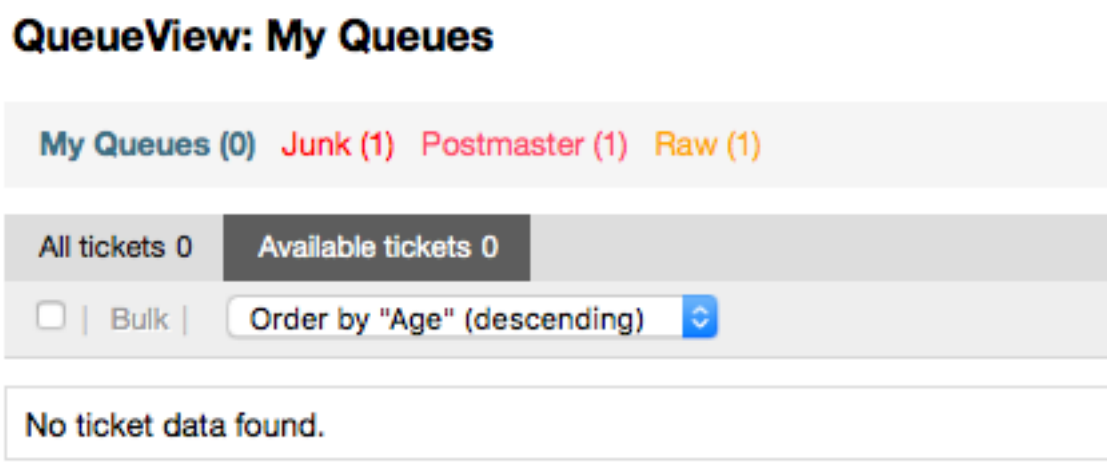
Sorting your incoming messages into different queues helps you to keep the support system structured and tidy. Because your agents are arranged into different groups with different access rights on queues, the system can be optimized even further. Queues can be used to define work flow processes or to create the structure of a company. Max could implement, for example, another queue called "sales", which could contain the sub queues "requests", "offers", "orders", "billing", etc. Such a queue structure could help Max to optimize his order transactions.

Improved system structures, such as through the proper design of queues, can lead to significant time and cost savings. Queues can help to optimize the processes in your company.

8. What is the Queue Overview?

The queue overview offers a view of all queues in which tickets are present, and for which the user has RW permissions.

Figure 3.10. Queue View (Default) for Agents



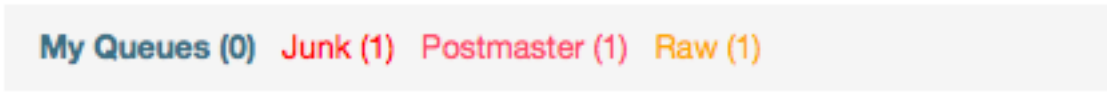
The queue overview offers a variety of options for daily work with OTRS. The first of these is the My Queue. In the Agent Preferences, or when administering agents, a set of queues can be defined for which the agent has been assigned to work within. All the tickets will appear in this default view, when accessing the Tickets -> Queue View menu.

The second option offered by the Queue View is a drill down navigation into individual queues and sub-queues containing tickets to be worked upon.

In both of the view types, the user also has the added ability to see either all unlocked tickets (this is the default filter), or the user can then choose to view all available tickets. Tickets must be in one of the viewable state types to be shown in the queue view. Per default, these are 'open, new, pending reminder, pending auto'.

There are visual alarms, to aid the user.

Figure 3.11. Agent Queue View visual alarms.



Visual Alarms

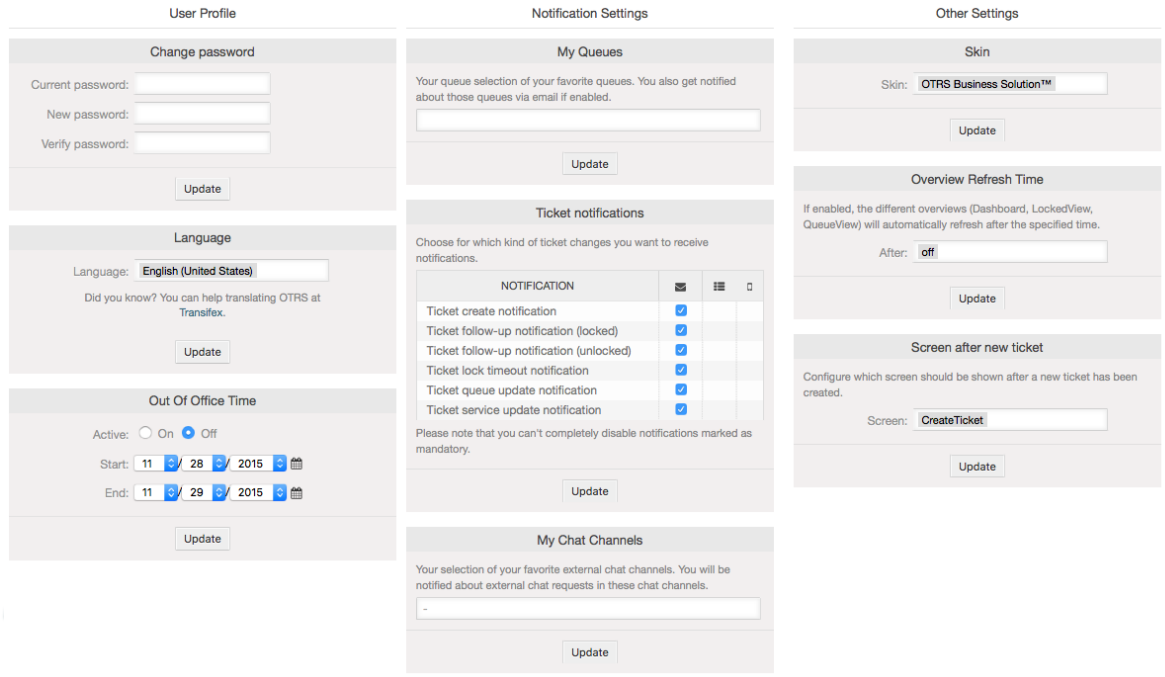
- Highlight Age 1: Sets the age in minutes (first level) for highlighting queues that contain untouched tickets. Seen in above in the "Raw" queue.
- Highlight Age 2: Sets the age in minutes (second level) for highlighting queues that contain untouched tickets. Seen in above in the "Postmaster" queue.
- Blink: Activates a blinking mechanism of the queue that contains the oldest ticket. Not supported in all browsers. In that case it will appear red, as seen in above in the "Junk" queue.
- Bold: The current queue will be bolded, as seen above in the "My Queues".

9. User Preferences

OTRS users such as customers, agents and the OTRS administrator can configure their account preferences as per their needs. Agent can access the configuration screen by

clicking on their login name at the top right corner of the web interface (see figure below), and customers must click on the "Preferences" link (see figure below).

Figure 3.12. Agent's personal preferences



An agent can configure 3 different categories of preferences: user profile, email settings, and other settings. The default possibilities are:

User Profile

- Change the current password.
- Adjust the interface language.
- Activate and configure the out-of-office time.
- Shift the frontend theme.

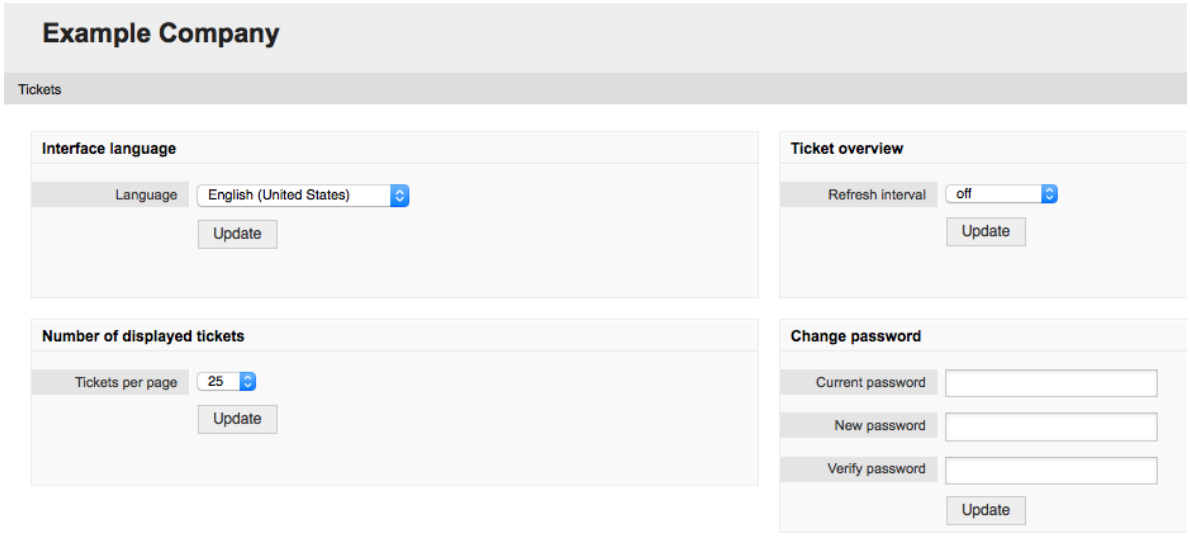
Notification Settings

- Select the queues you want to monitor in "My Queues".
- Select the services you want to monitor in "My Services".
- Configure which ticket notifications you want to receive (per transport method).

Other Settings

- Switch the frontend skin.
- Set the refresh period for the overviews (Dashboard, Queue View, etc.).
- Set the screen to be displayed after a ticket is created.

Figure 3.13. Customer's personal preferences



A customer can select the web interface language, set the refresh interval for the ticket overview, and choose the maximum amount of shown tickets. It is also possible to set a new password.

Chapter 4. Administration

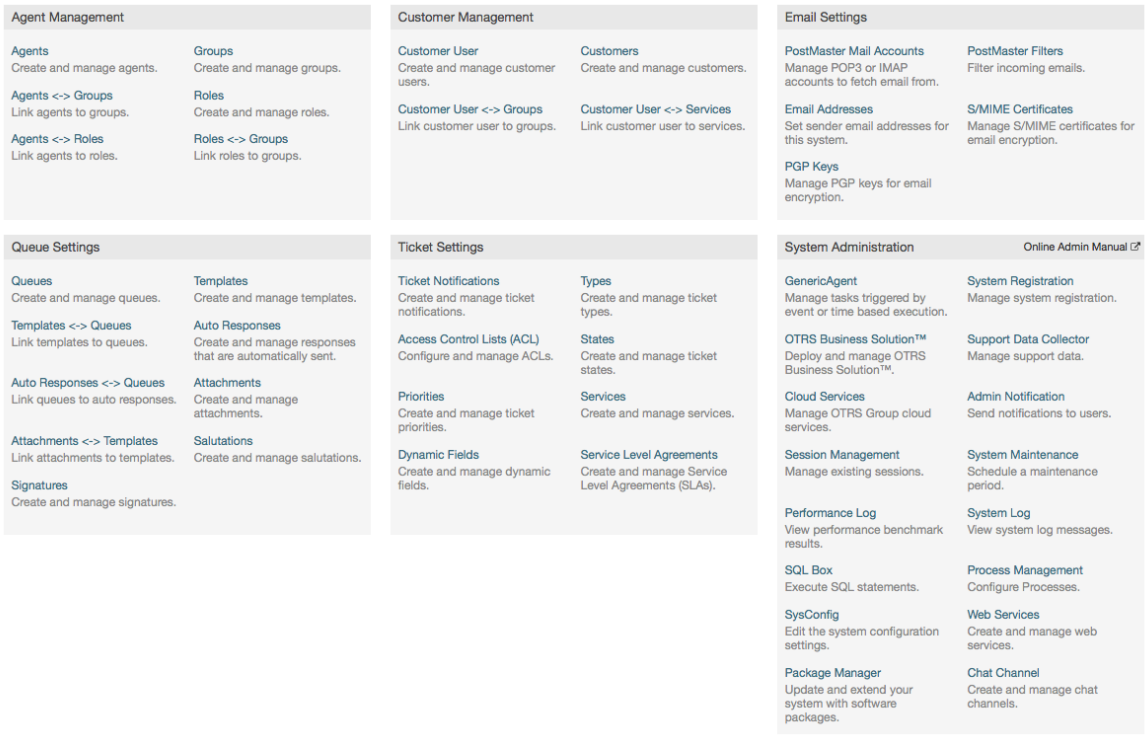
1. The Administration Area of OTRS

1.1. Basics

The following system configuration settings are available to OTRS administrators by accessing the Admin page of the OTRS web interface - adding agents, customers and queues, ticket and mail settings, installing additional packages such as FAQ and ITSM, and much more.

Agents who are members of the *admin* group can access the Admin area by clicking the *Admin* link in the navigation bar (see figure below). Agents without sufficiently elevated access rights will not be able to access this link.

Figure 4.1. OTRS Administration Overview Screen



1.2. Agents, Groups and Roles

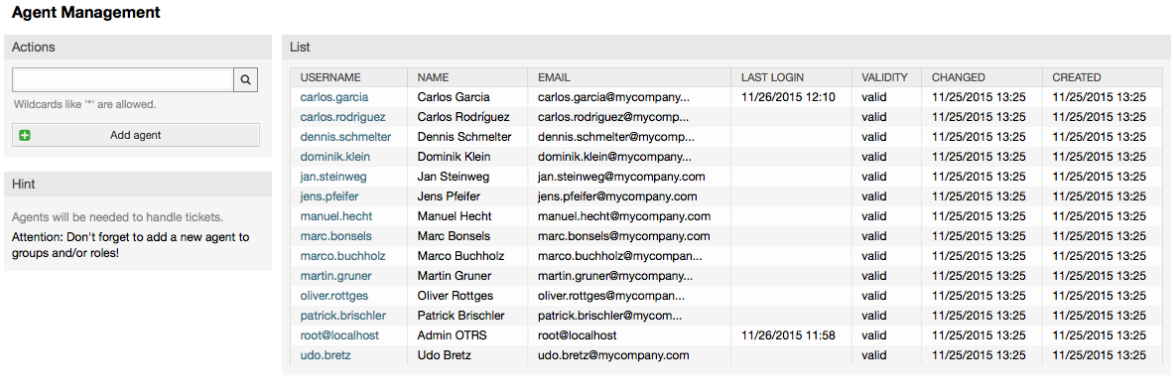
1.2.1. Agents

By clicking the link *Agents*, you get access to the agent management screen of OTRS (see figure below). Administrators can add, change or deactivate agent accounts. Furthermore they can also manage agent preferences, including the language and notification settings for the individual agent's interface.

Note

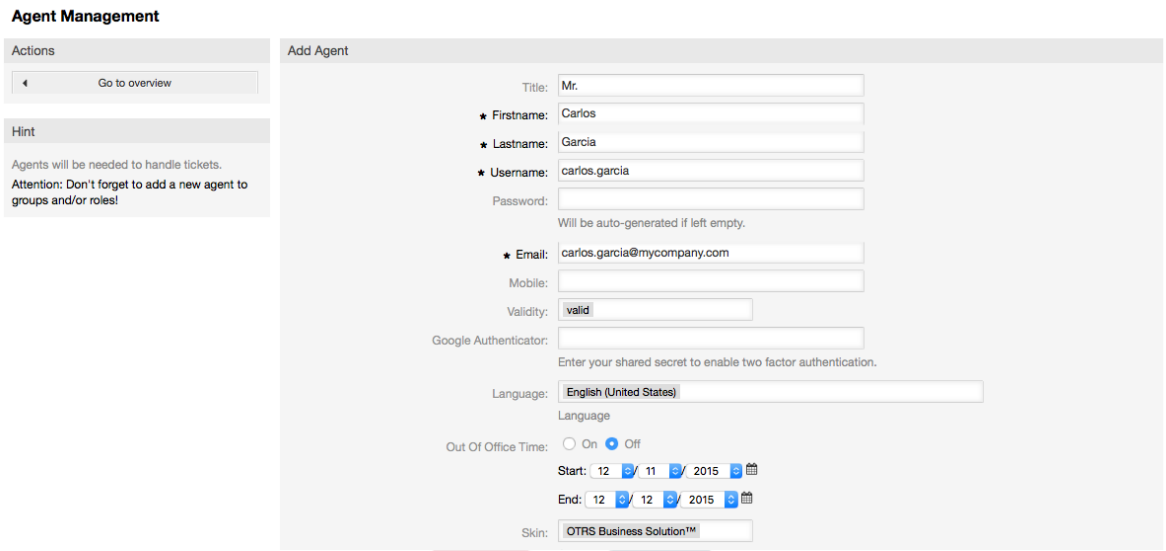
An OTRS agent account may be deactivated but not deleted. Deactivation is done by setting the Valid flag to *invalid* or *invalid-temporarily*.

Figure 4.2. Agent Management



To register an agent, click on the "Add agent" button, enter the required data and press the Submit button at the bottom of the screen, as shown in Figure.

Figure 4.3. Adding a new agent



After the new agent account has been created, you should make the agent a member of one or more groups or roles. Information about groups and roles is available in the Groups and Roles sections of this chapter.

1.2.2. Groups

Every agent's account should belong to at least one group or role. In a brand new installation, there are three pre-defined groups available, as shown in Table 4-1.

Table 4.1. Default groups available on a fresh OTRS installation

Group	Description
admin	Allowed to perform administrative tasks in the system.
stats	Qualified to access the stats module of OTRS and generate statistics.

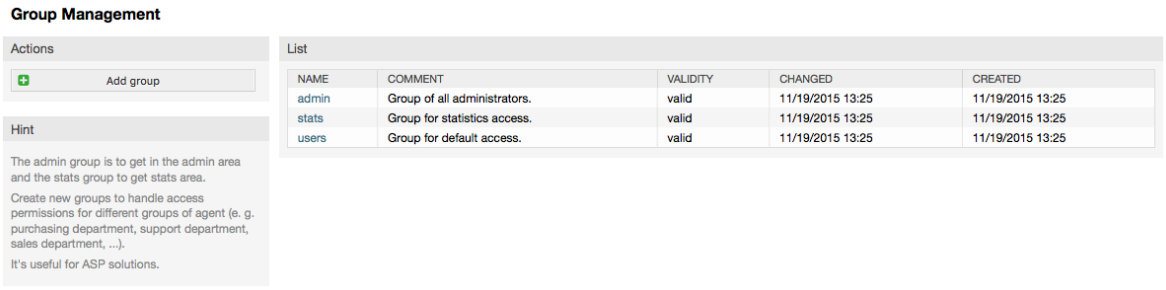
Group	Description
users	Agents should belong to this group, with read and write permissions. They can then access all functions of the ticket system.

Note

In a brand new OTRS installation, the *users* group initially does not have any members. The agent 'root@localhost' belongs by default to the admin and stats groups.

You can access the group management page (see figure below) by clicking the *Groups* link in the admin area.

Figure 4.4. Group management



Note

As with agents, an OTRS group may be deactivated but not deleted. Deactivation is done by setting the Valid flag to *invalid* or *invalid-temporarily*.

To add an agent to a group, or to change the agents who belong to a group, you can use the link *Agents <-> Groups* from the Admin page (see figure below).

Figure 4.5. Agent <-> group management

An overview of all groups and agents in the system is displayed on this page. You can also use the available filters to find a specific entity. If you want to change the groups that an agent is a member of, just click on the agent's name (see figure below). To change the agents associated with a group, just click on the group you want to edit (see figure below).

Figure 4.6. Change the groups an agent belongs to

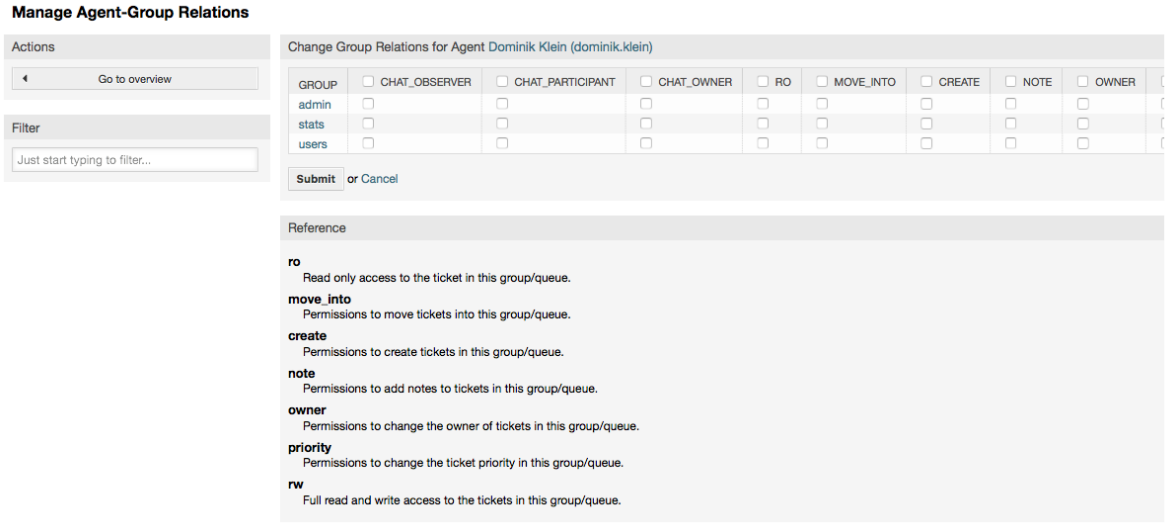
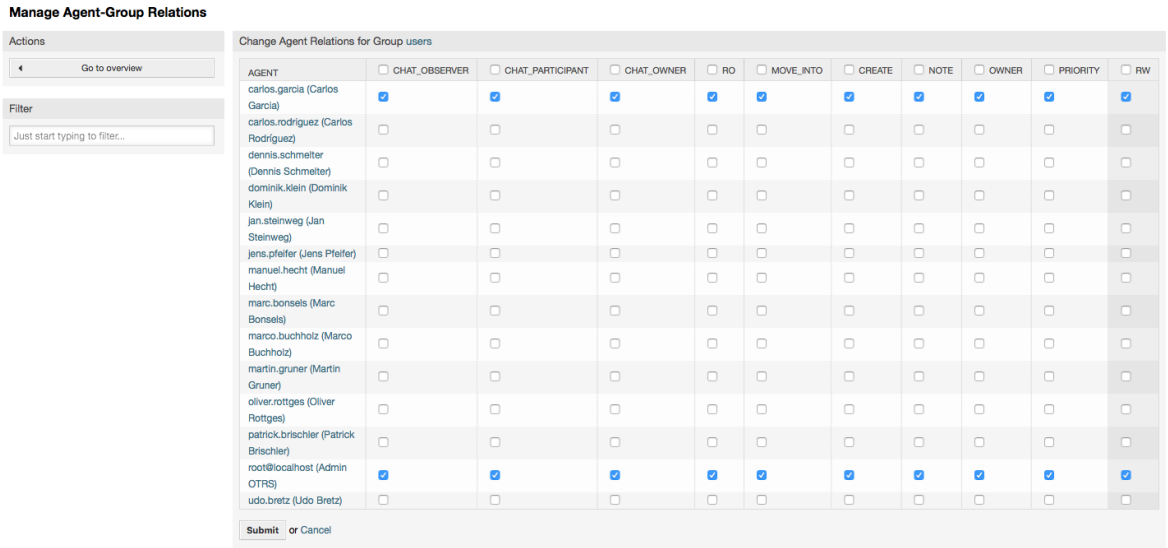


Figure 4.7. Change the agents that belong to a specific group



Each group has a set of rights associated with it, and each group member (agent) may have some combination of these rights for themselves. A list of the permissions / rights is shown in Table 4-2.

Table 4.2. Rights associated with OTRS groups

Right	Description
chat_observer	Agents may take part silently in a chat (available in OTRS Business Solution™).
chat_participant	Agents may normally participate in a chat (available in OTRS Business Solution™).
chat_owner	Agents have full rights for a chat and can accept chat requests (available in OTRS Business Solution™).
ro	Read only access to the tickets, entries and queues of this group.

Right	Description
move into	Right to move tickets or entries between queues or areas that belong to this group.
create	Right to create tickets or entries in the queues or areas of this group.
owner	Right to update the owner of tickets or entries in queues or areas that belong to this group.
priority	Right to change the priority of tickets or entries in queues or areas that belong to this group.
rw	Full read and write access on tickets or entries in the queues or areas that belong to this group.

Note

By default, the QueueView only lists tickets in queues that an agent has *rw* access to, i.e., the tickets the agent needs to work on. If you want to change this behaviour, you can set Ticket::Frontend::AgentTicketQueue###ViewAllPossibleTickets to *Yes*.

Not all available permissions are shown by default. These additional permissions can be added.

Table 4.3. Additional permission groups

Right	Description
stats	Gives access to the stats page.
bounce	The right to bounce an email message (with bounce button in ticketZoom).
compose	The right to compose an answer for a ticket.
customer	The right to change the customer of a ticket.
forward	The right to forward a messages (with the forward button).
pending	The right to set a ticket to pending.
phone	The right to add a phonecall to a ticket.
responsible	The right to change the responsible agent for a ticket.

Note

These permissions can be added by changing the System::Permission

1.2.3. Roles

Roles are a powerful feature to manage the access rights of many agents in a very simple and quick manner. They are particularly useful for large, complex support systems with a lot of agents, groups and queues. An example below explains when they should be used.

Suppose that you have a system with 100 agents, 90 of them with access to a single queue called "support" where all support requests are handled. The "support" queue contains

multiple sub queues. The other 10 agents have permission to access all queues of the system. These 10 agents dispatch tickets, watch the raw queue and move spam messages into the "junk" queue.

The company now opens a new department that sells some products. Order request and acceptance, order confirmation, bills, etc. must be processed, and some of the company's agents are supposed to do this using OTRS. The different agents have to get access to the new queues that must be created.

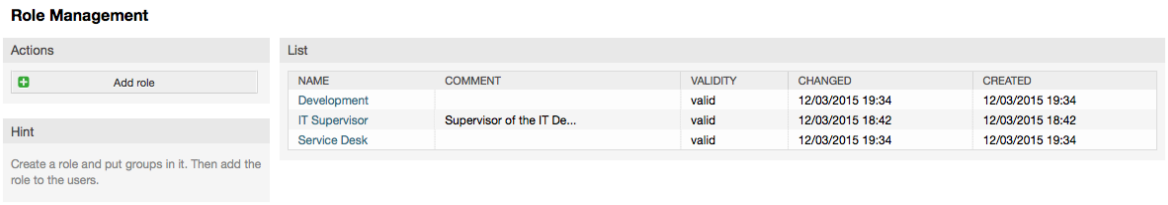
Because it would take a long time to change the access rights for the individual agents manually, roles that define the different access levels can be created. The agents can then be added to one or more roles, with their access rights being modified automatically. If a new agent account is created, it is also possible to add this account to one or more roles.

Note

Roles are really useful when dealing with complex organizations and when maintaining larger OTRS installations. Proper care is advised though. Mixing Agent to Group with Agent to Role mappings can make for a complex access control scheme, that is difficult to understand and maintain. If you wish to use only roles and disable the Agents <-> Groups option in the Admin area, you can do so by modifying the Frontend::Module###AdminUserGroup in the SysConfig. Be aware that this won't remove already existing Agents to Group assignments!

You can access the role management section (see figure below) by clicking the *Roles* link on the Admin page.

Figure 4.8. Role management

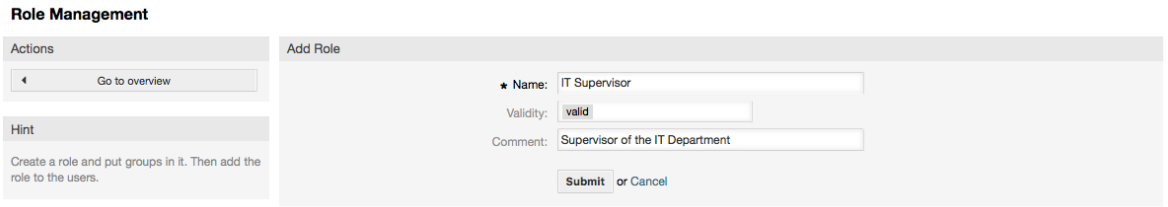


Note

As with agent and groups, roles once created can be deactivated but not deleted. To deactivate, set the Valid option to *invalid* or *invalid-temporarily*.

An overview of all roles in the system is displayed. To edit a role's settings, click on the role's name. In a fresh new OTRS installation, there are no roles defined by default. To register one, click on the "Add role" button, provide the needed data and submit it (see figure below).

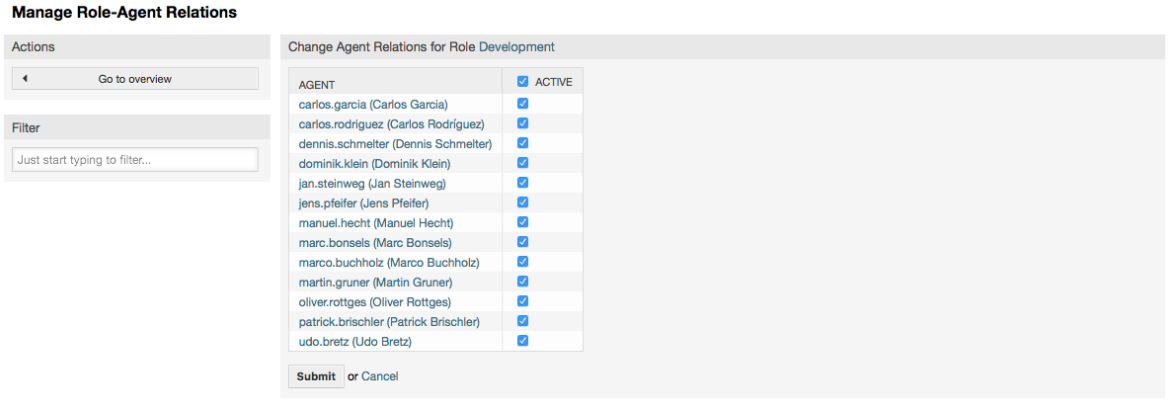
Figure 4.9. Adding a new role



To get an overview of all roles and agents in the system, click on the link Roles <-> Agents on the Admin page. You can also use filters to find a specific element. If you want to change the roles associated with an agent, just click on the agent's name (see figure below). To change the agents associated with a role, click on the role you want to edit (see figure below).

Figure 4.10. Change the roles associated with an agent

Figure 4.11. Change the agents associated with a specific role



To get an overview of all roles and groups in the system, click on the link Roles <-> Groups on the Admin page. You will see a similar screen as the one shown in the Figure. You can also use filters to find a specific entity.

Figure 4.12. Manage roles-groups relations

To define the different access rights for a role, click on the name of a role or a group (see below the Figures 4.13 and 4.14, respectively).

Figure 4.13. Change group relations for a role

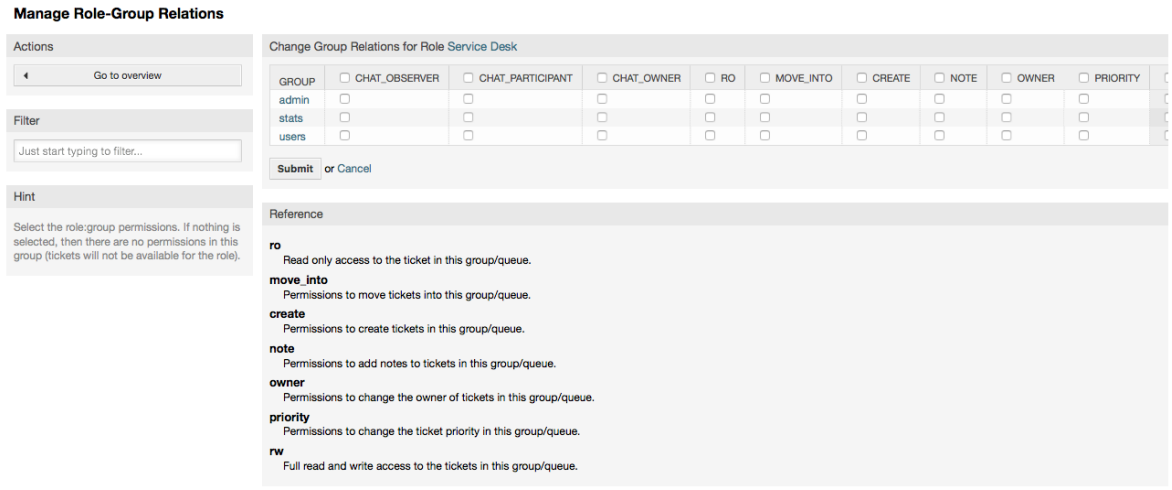
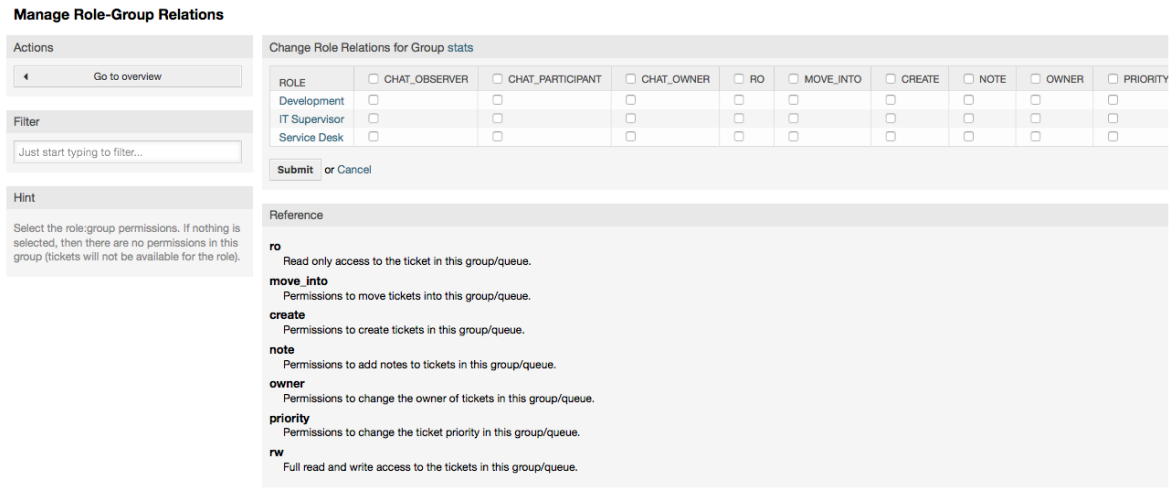


Figure 4.14. Change role relations for a group

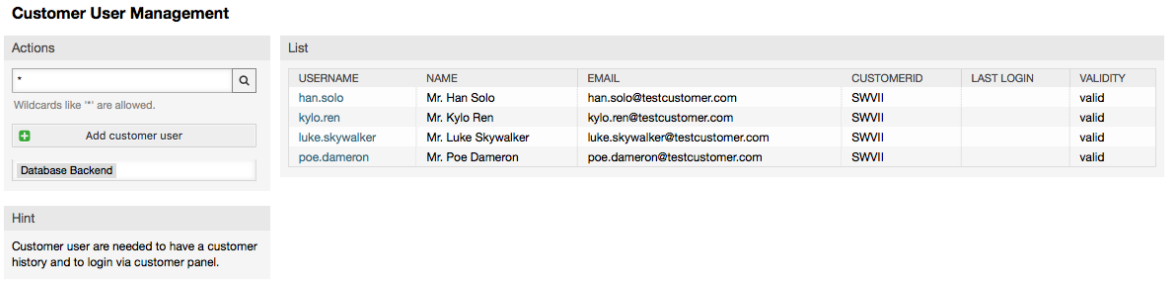


1.3. Customers and Customer Groups

1.3.1. Customers

OTRS supports different types of users. Using the link "Customers" (via the navigation bar, or the Admin page), you can manage the accounts of your customers (see figure below), who can log into the system via the Customers interface (customer.pl). Through this interface, your customers can not only create tickets but also review their past tickets for new updates. It is important to know that a customer is needed for the ticket history in the system.

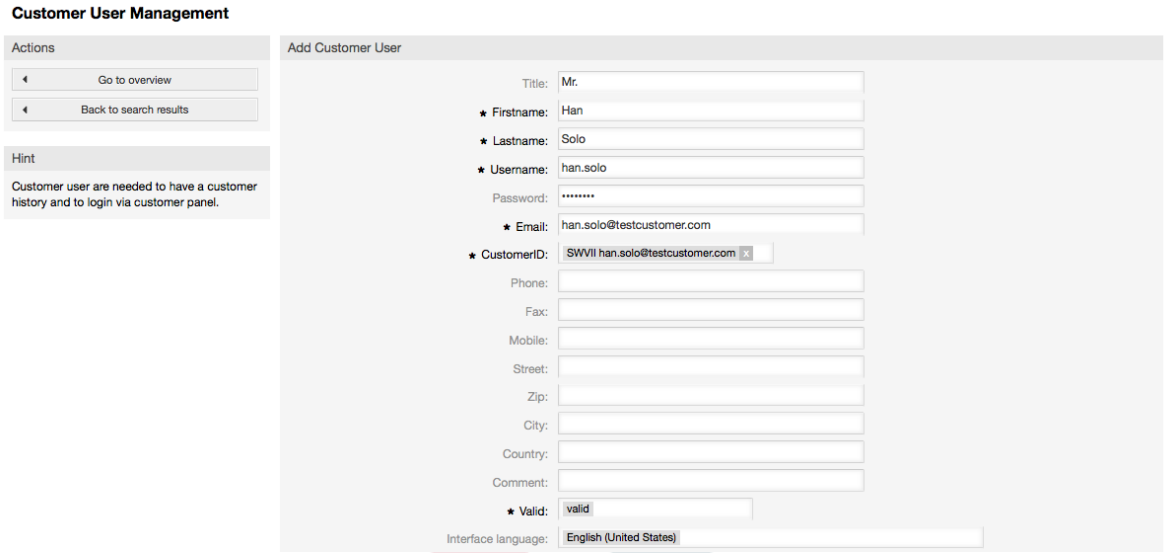
Figure 4.15. Customer management



You can search for a registered customer, or edit their settings by clicking on their name. You also have the possibility to change the customer back-end, for further information please refer to the chapter about external back-ends.

To create a new customer account, click on the "Add customer" button (see figure below). Some of the fields are mandatory, i.e., they have to contain values, so if you leave one of those empty, it will be highlighted in red.

Figure 4.16. Adding a customer



Customers can access the system by providing their username and password. The CustomerID is needed by the system to identify the user and associated tickets. Since the email address is a unique value, it can be used as the ID.

Note

As with agents, groups and roles, customers can not be deleted from the system, only deactivated by setting the Valid option to *invalid* or *invalid-temporarily*.

1.3.2. Customer Groups

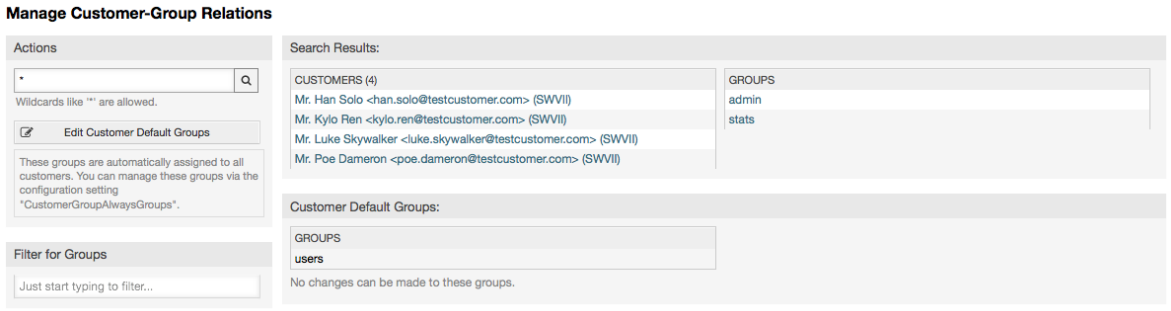
Customer users can also be added to a group, which can be useful if you want to add customers of the same company with access to one or a few queues. First create the group to which your customers will belong, via the Group management module. Then add the queues and select the new group for the queues.

The next step is to activate the customer group support. This can be done with the configuration parameter CustomerGroupSupport, from the Admin SysConfig option. Using the

parameter CustomerGroupAlwaysGroups, you can specify the default groups for a newly added customer, so that every new account will be automatically added to these groups.

Through the link "Customers <-> Groups" you can manage which customer shall belong to the different groups (see figure below).

Figure 4.17. Customer-Group relations management



To define the different groups a customer should be part of and vice versa, click on the corresponding customer username or group (see below the Figures 4.18 and 4.19, respectively).

Figure 4.18. Change Group relations for a Customer

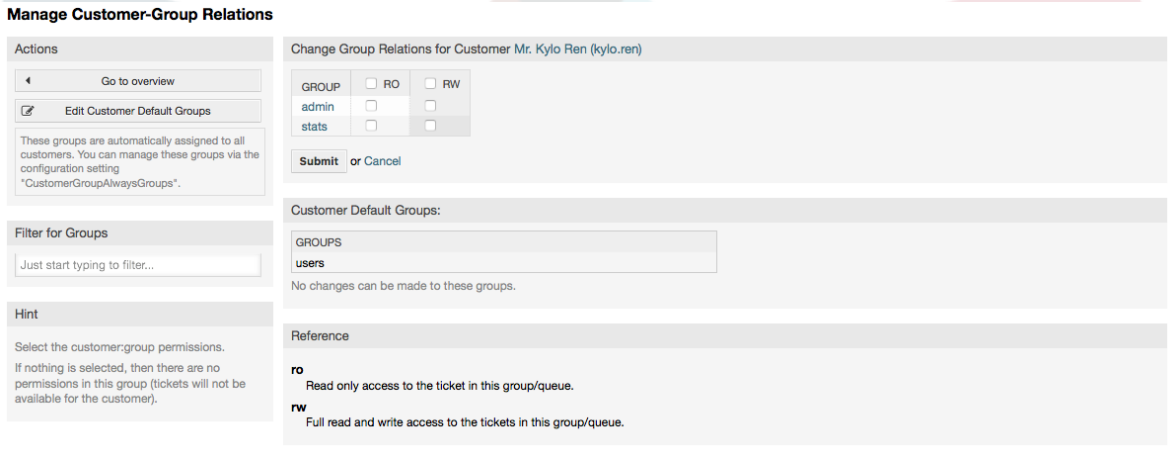
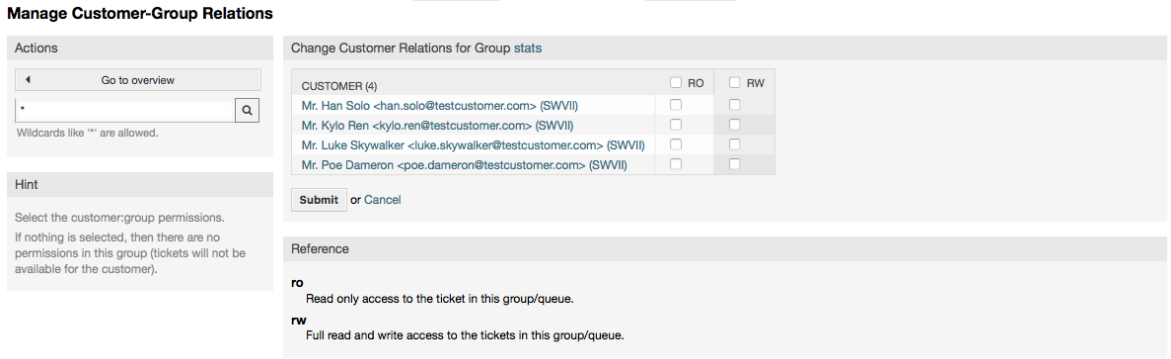


Figure 4.19. Change Customer relations for a Group

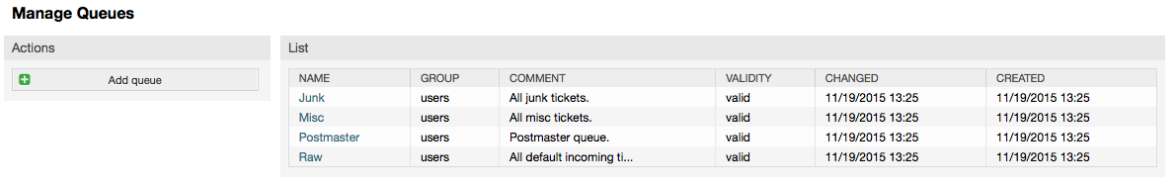


1.4. Queues

Clicking on the link "Queues" of the Admin page, you can manage the queues of your system (see figure below). In a new OTRS installation there are 4 default queues: Raw,

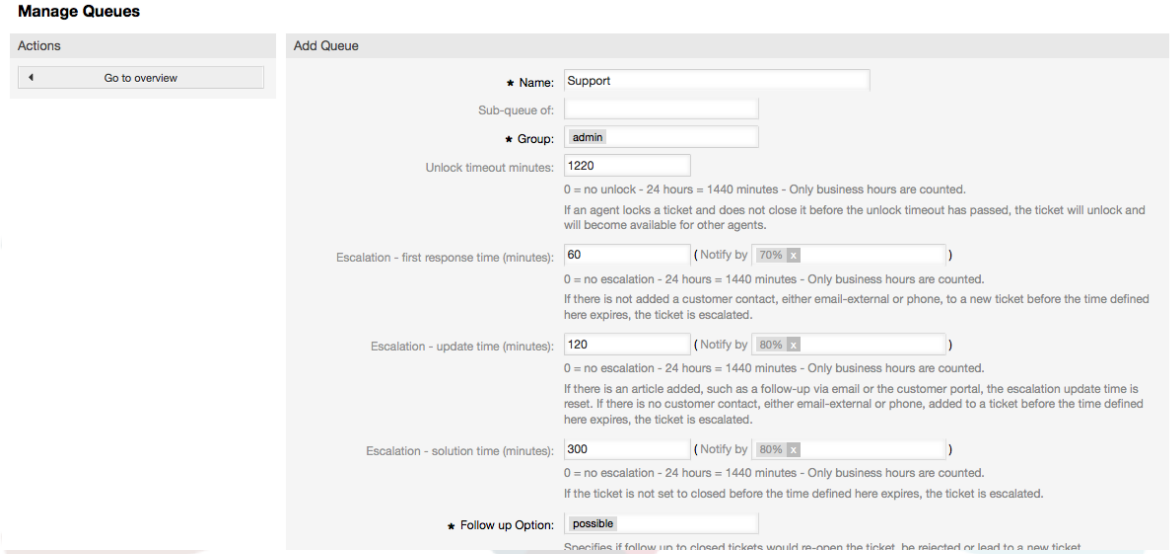
Junk, Misc and Postmaster. All incoming messages will be stored in the "Raw" queue if no filter rules are defined. The "Junk" queue can be used to store spam messages.

Figure 4.20. Queue management



Here you can add queues (see figure below) and modify them. You can specify the group that should use the queue. You can also set the queue as a sub-queue of an existing queue.

Figure 4.21. Adding a new queue



You can define an unlock timeout for a queue - if an agent locks a ticket and does not close it before the unlock timeout has passed, the ticket will be automatically unlocked and made available for other agents to work on.

There are three escalation time settings that can be associated at queue level:

Escalation - First Response Time

- After creation of the ticket, if the time defined here expires without any communication with the customer, either by email or phone, the ticket is escalated.

Escalation - Update Time

- If there is a customer followup either via e-mail or the customer portal, that is recorded in the ticket, the escalation update time is reset. If there is no customer contact before the time defined here expires, the ticket is escalated.

Escalation - Solution Time

- If the ticket is not closed before the time defined here expires, the ticket is escalated.

With 'Ticket lock after a follow-up', you can define if a ticket should be set to 'locked' to the old owner if a ticket that has been closed and later is re-opened. This ensures that a follow up for a ticket is processed by the agent that has previously handled that ticket.

The parameter for the system address specifies the email address that will be used for the outgoing tickets of this queue. There is also the possibility to associate a queue with a salutation and a signature, for the email answers. For more detailed information, please refer to the sections email addresses, salutations and signatures.

Note

As with agents, groups and customers, queues cannot be deleted, only deactivated, by setting the Valid option to *invalid* or *invalid-temporarily*.

1.5. Salutations, Signatures, Attachments and Templates

1.5.1. Salutations

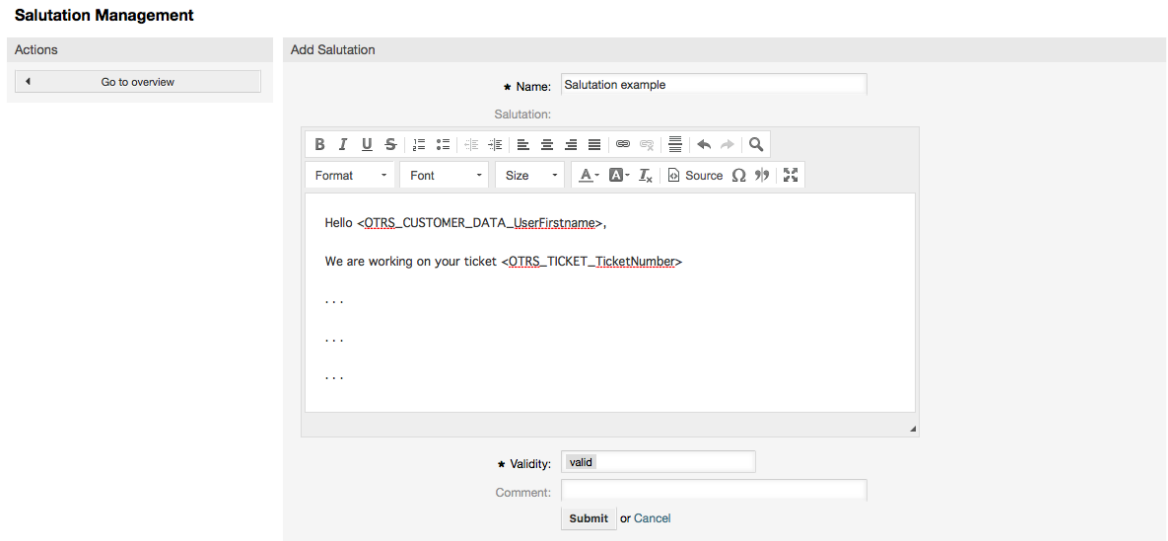
A salutation is a text module for a template. Salutations can be linked to one or more queues, as described in the section about queues. A salutation is used only if a ticket from a queue the salutation is linked to, is answered. To manage the different salutations of your system, use the "Salutations" link of the admin area (see figure below).

Figure 4.22. Salutation management

After a default installation there is already one salutation available, "system standard salutation (en)".

To create a new salutation, press the button "Add salutation", provide the required data and submit it (see figure below).

Figure 4.23. Adding a new salutation



It is possible to use variables in salutations. When you respond to a ticket, the variable names will be replaced by their values.

The different variables you can use in templates are listed in the lower part of the salutation screen. If you use, for example, the variable <OTRS_LAST_NAME> the last name of the ticket's sender will be included in your reply.

Note

As with other OTRS entities, salutations cannot be deleted, only deactivated by setting the Valid option to *invalid* or *invalid-temporarily*.

1.5.2. Signatures

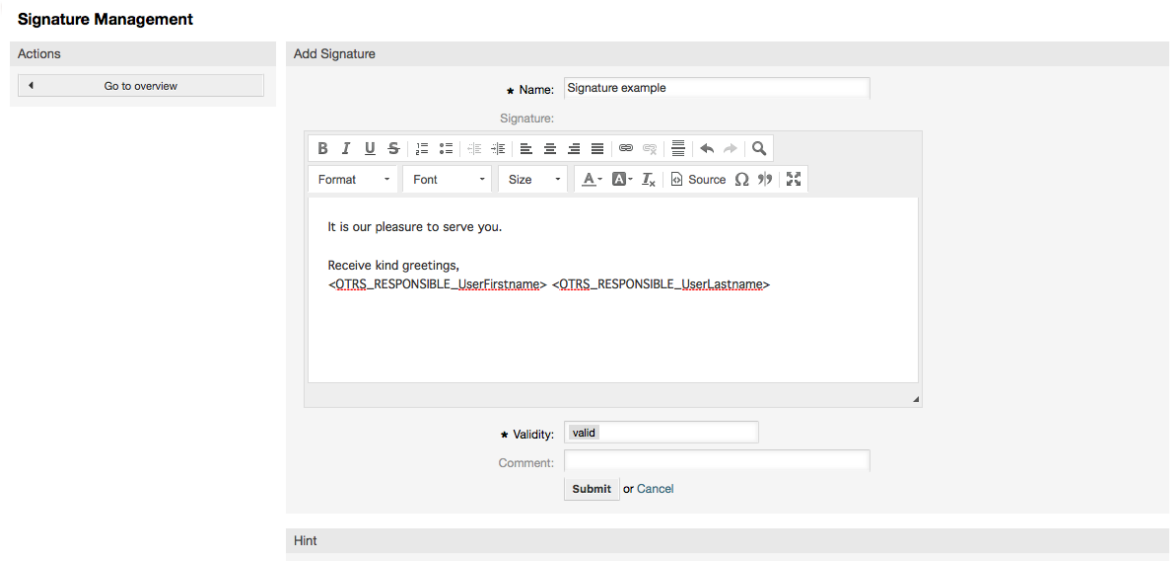
Another text module for a template is the signature. Signatures can be linked to a queue, as described in the section about the queues. Please note that a signature will only be appended to a template text, if it has previously been linked to a queue. You can manage the signatures in your system by accessing the "Signatures" link of the Admin page, (see figure below).

Figure 4.24. Signatures management

After a fresh installation of OTRS, there is one predefined signature stored in your system, "system standard signature (en)".

To create a new signature, press the button "Add signature", provide the needed data and submit it (see figure below).

Figure 4.25. Adding a new signature



Like salutations, signatures can also contain dynamic content, such as the first and last name of the agent who answers the ticket. Here too, variables can be used to replace the content of the signature text for every ticket. See the lower part of the signatures screen for the variables which can be used. If you include the variable <OTRS_LAST_NAME> in a signature for example, the last name of the agent who answers the ticket will replace the variable.

Note

As with salutations, signatures too cannot be deleted, only deactivated by setting the Valid option to *invalid* or *invalid-temporarily*.

1.5.3. Attachments

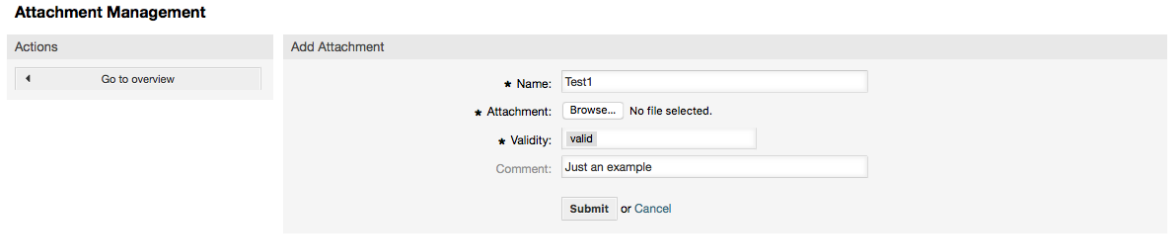
You can also optionally add one or more attachments to a template. If the template is selected, the attachments will be attached to the message composition window. If necessary, the agent can remove the attachment from an individual template before sending it to the customer.

Through the "Attachment" link of the Admin page, you can load the attachments into the database of the system (see figure below).

Figure 4.26. Attachments management

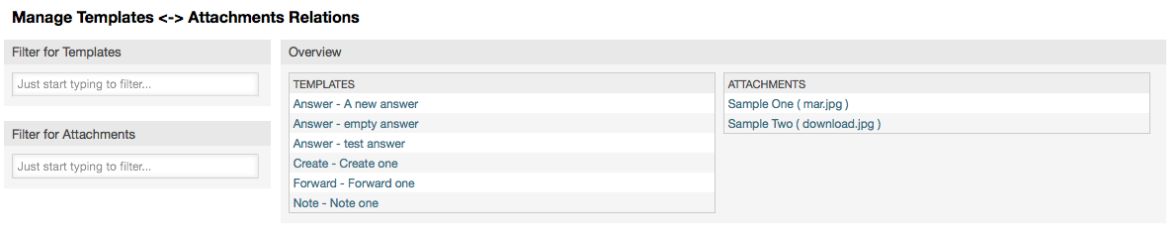
To create a new attachment, press the button "Add attachment", provide the required data and submit it (see figure below).

Figure 4.27. Adding a new attachment



If an attachment is stored it can be linked to one or more templates. Click on the "Attachment <-> Templates" link of the Admin page (see figure below).

Figure 4.28. Linking Attachments to Templates



To associate different attachments with a specific template and vice versa, click on the corresponding template name or attachment (see below the Figures 4.29 and 4.30, respectively).

Figure 4.29. Change Attachment relations for a Template

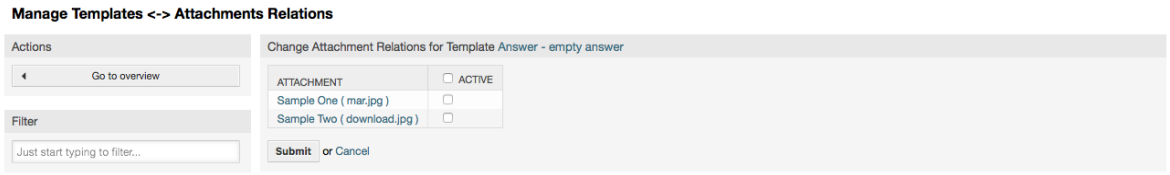
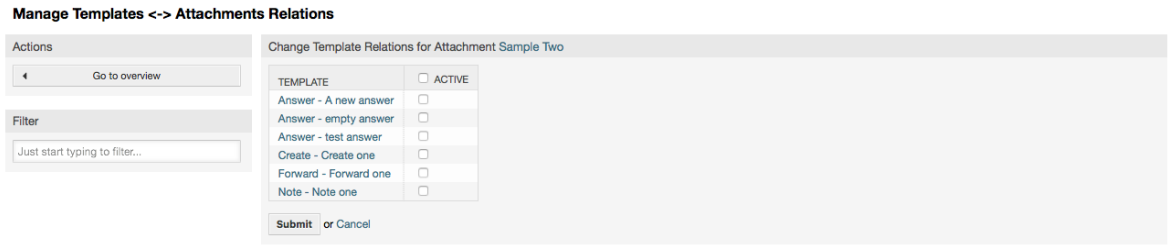


Figure 4.30. Change Template relations for an Attachment



1.5.4. Templates

To speed up ticket processing and to standardize the look of answers, you can define templates in OTRS. A template can be linked to one or more queues and vice versa.

There are different kind of templates that are used in different parts of OTRS and they have its own purpose, the following is the list of possible template types:

- Answer: To be used as a ticket response or reply
- Create: To be used in New Phone or Email ticket
- Forward: To be used to forward an article to someone else
- PhoneCall: To be used in the Phone Call Inbound and Outbound screens

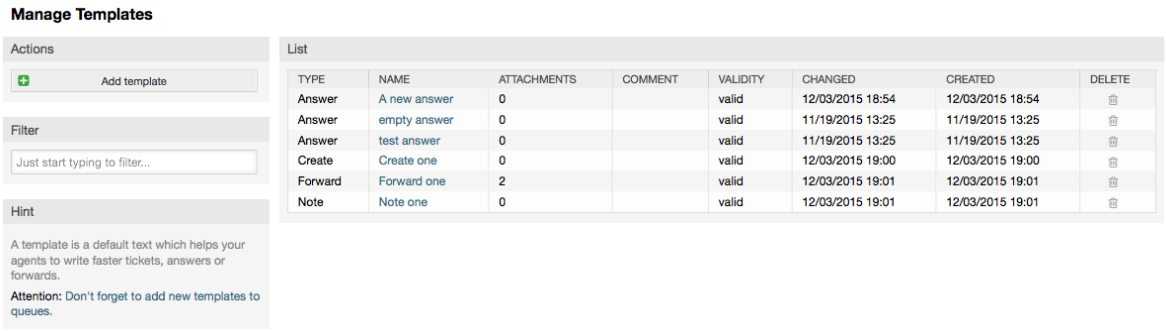
Answer templates can be accessed in two ways, from the ticket zoom screen in the article menu, or on a quicker way: from any ticket overview large screen such as Status View or Ticket View. For a fresh OTRS installation, the "empty answer" template (Answer) is set as the default for every queue.

As soon as Forward templates are added and assigned to queues, the "Forward" button in ticket zoom (that normally leads to a empty text forward screen) will change into a selection control, the selection is filled with the added Forward templates, by choosing one of the templates, the forward screen will be shown prefilled with the template text and attachments (similar to the reply selection box with the Answer templates).

Creating templates of type Create and PhoneCall will make visible the "Text Template" selection box in their respective screens, choosing a template for the list will populate the "Text" and "Attachment" fields (if available in the template). Notice that any previous change in the Text or attachments will be overwritten by selecting a template.

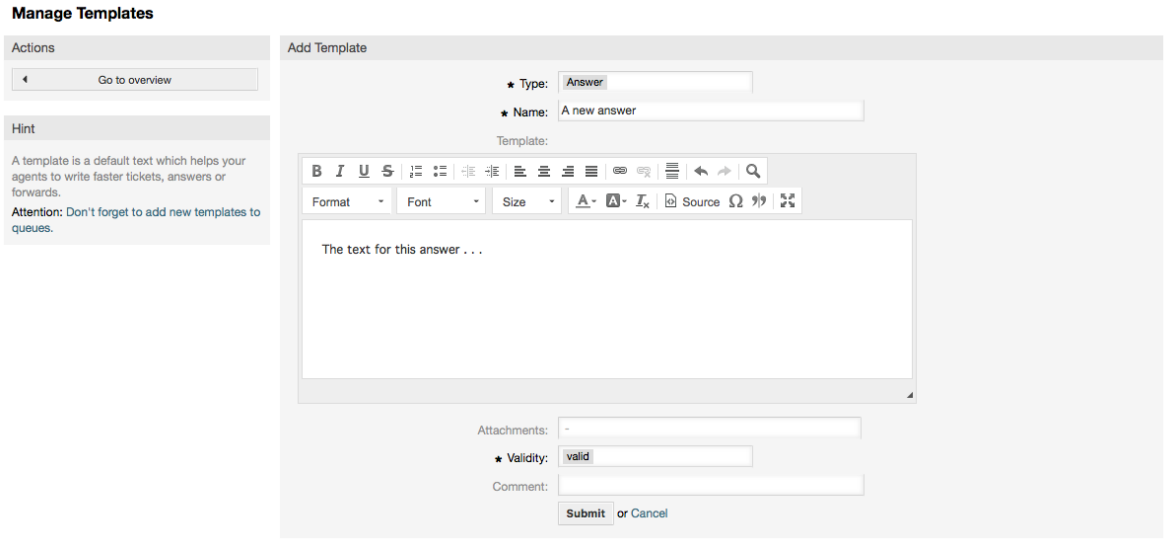
Clicking the "Templates" link on the Admin page brings you to the Template management screen (see figure below).

Figure 4.31. Template management



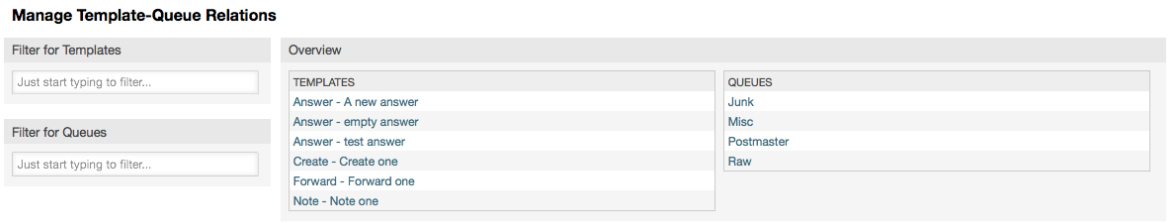
To create a new template, click on the "Add template" button, provide the required data (make sure to select the appropriate template type) and submit it (see figure below).

Figure 4.32. Adding a template



To add/remove templates to one or more queues, click on the "Templates <-> Queues" link on the Admin page (see figure below). You can also use filters to get information regarding a specific entity.

Figure 4.33. Template-Queue relations management



To define the different templates that will be available for a queue and vice versa, click on the corresponding template or queue (see below the Figures 5.32 and 5.33, respectively).

Figure 4.34. Change Queue relations for a Template

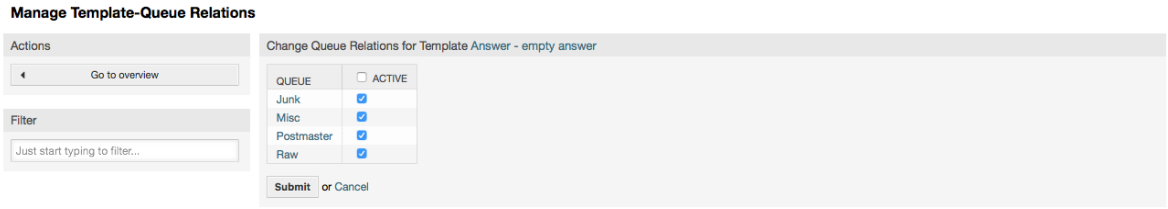


Figure 4.35. Change Template relations for a Queue

When choosing a template, additional information could be added to the template text, this depends on the template type:

PhoneCall and Create templates does not add any content to the template text, however New Email Ticket screen adds the queue assigned signature to the resulting email body (this screen has a separated box to visualize the signature).

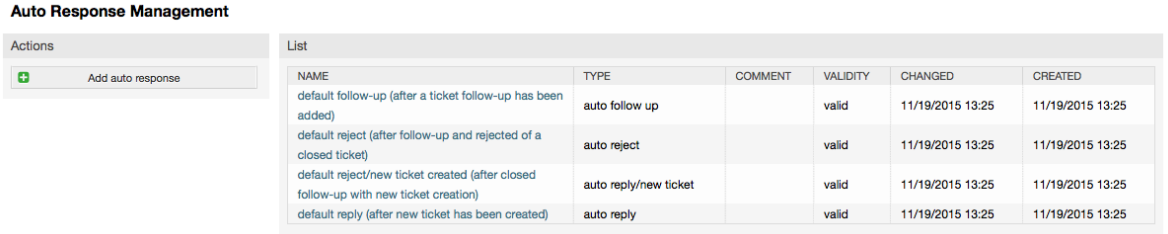
Answer templates text when selected also included the salutation associated with the queue, then the text of the template, then the quoted ticket text, and finally the signature associated with the queue.

Forward templates are similar to Answer templates, but they does not include the salutation part.

1.6. Auto Responses

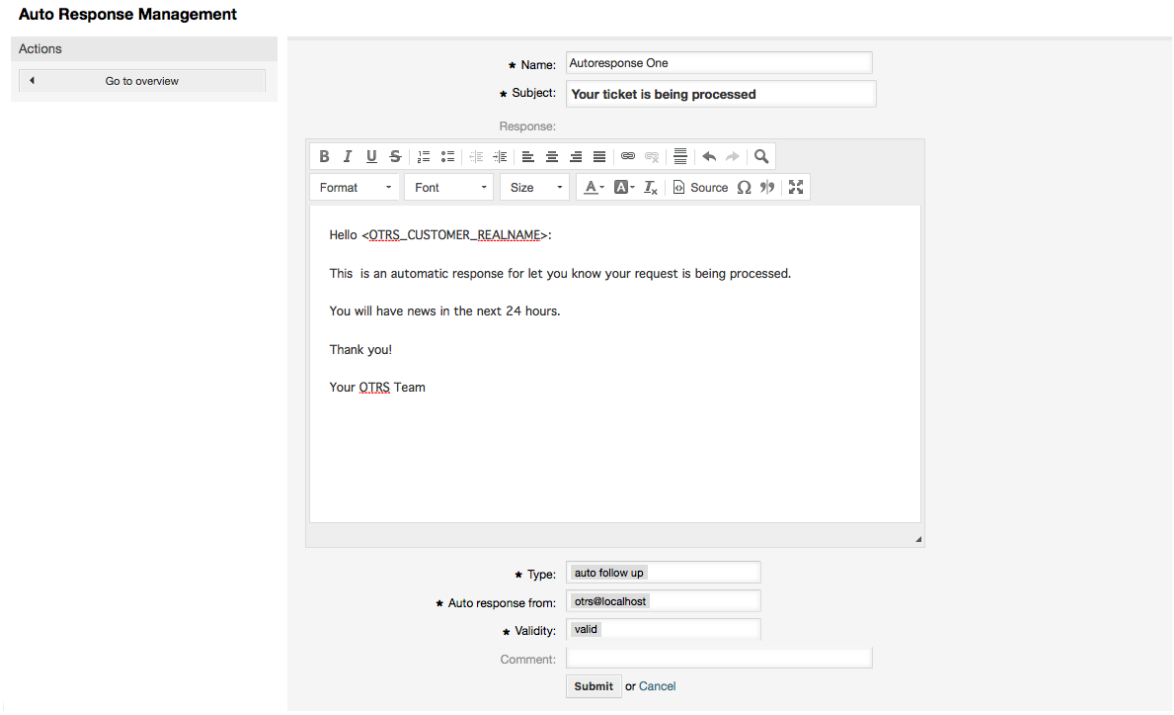
OTRS allows you to send automatic responses to customers based on the occurrence of certain events, such as the creation of a ticket in a specific queue, the receipt of a follow-up message in regards to a ticket, the closure or rejection of a ticket, etc. To manage such responses, click the link "Auto responses" on the Admin page (see figure below).

Figure 4.36. Auto response management



To create an automatic response, click on the button "Add auto response", provide the needed data and submit it (see figure below).

Figure 4.37. Adding an auto response



The subject and text of auto responses can be generated by variables, just as in signatures and salutations. If you insert, for example, the variable <OTRS_CUSTOMER_EMAIL[5]> into the body of the auto answer, the first 5 lines of the customer mail text will be inserted into the auto answer. You will find more details about the valid variables that can be used at the bottom of the screen shown in the Figure.

For every automatic answer, you can specify the event that should trigger it. The system events that are available after a default installation are described in the Table 4-4.

Table 4.4. Events for auto responses

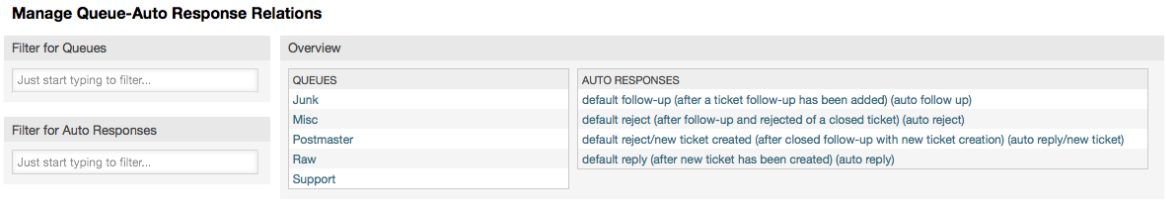
Name	Description
auto reply	Creation of a ticket in a certain queue.
auto reply/new ticket	Reopening of an already closed ticket, e.g. if a customer replies to such ticket.
auto follow up	Reception of a follow-up for a ticket.
auto reject	Automatic rejection of a ticket, done by the system.
auto remove	Deletion of a ticket, done by the system.

Note

As with other OTRS entities, auto responses too cannot be deleted, only deactivated, by setting the Valid option to *invalid* or *invalid-temporarily*.

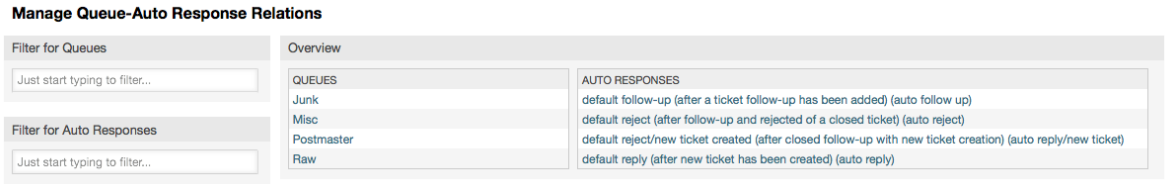
To add an auto response to a queue, use the "Auto Response <-> Queues" link on the Admin page (see figure below). All system events are listed for every queue, and an auto answer with the same event can be selected or removed via a listbox.

Figure 4.38. Queue <-> auto response relations management



To define the different auto responses that will be available for a queue, click on the corresponding queue name (see figure below). It is also possible to edit an existing auto response - to do so, click on the response and edit in the same manner as editing a new auto response.

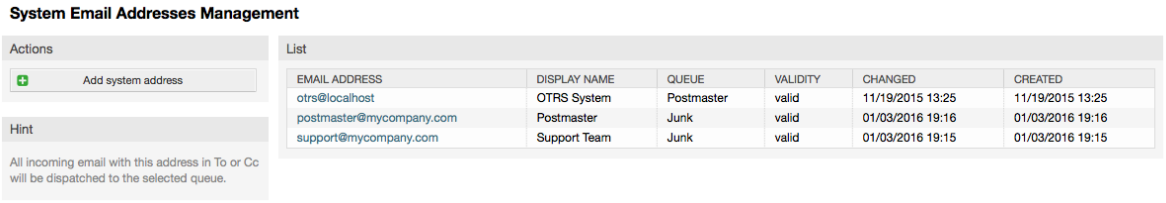
Figure 4.39. Change auto response relations for a queue



1.7. System Email Addresses

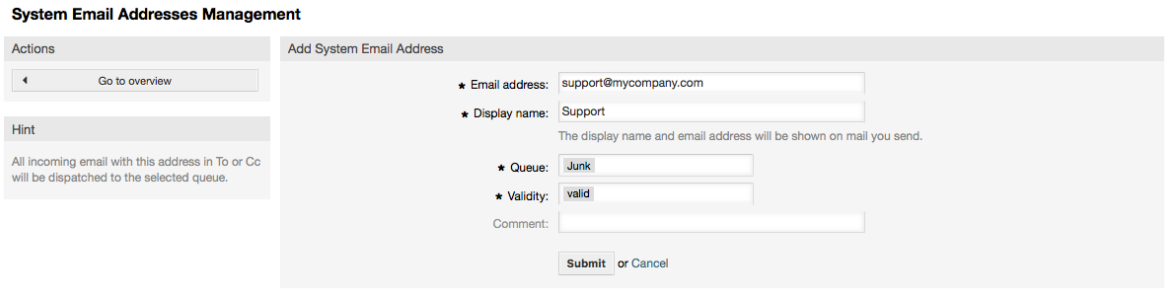
To enable OTRS to send emails, you need a valid email address to be used by the system. OTRS is capable of working with multiple email addresses, since many support installations need to use more than one. A queue can be linked to many email addresses, and vice versa. The address used for outgoing messages from a queue can be set when the queue is created. Use the "Email Addresses" link from the Admin page to manage all email addresses of the system (see figure below).

Figure 4.40. System email addresses management



If you create a new mail address (see figure below),you can select the queue or sub queue to be linked with it. This link enables the system to sort incoming messages via the address in the To: field of the mail into the right queue.

Figure 4.41. Adding a system email address



Note

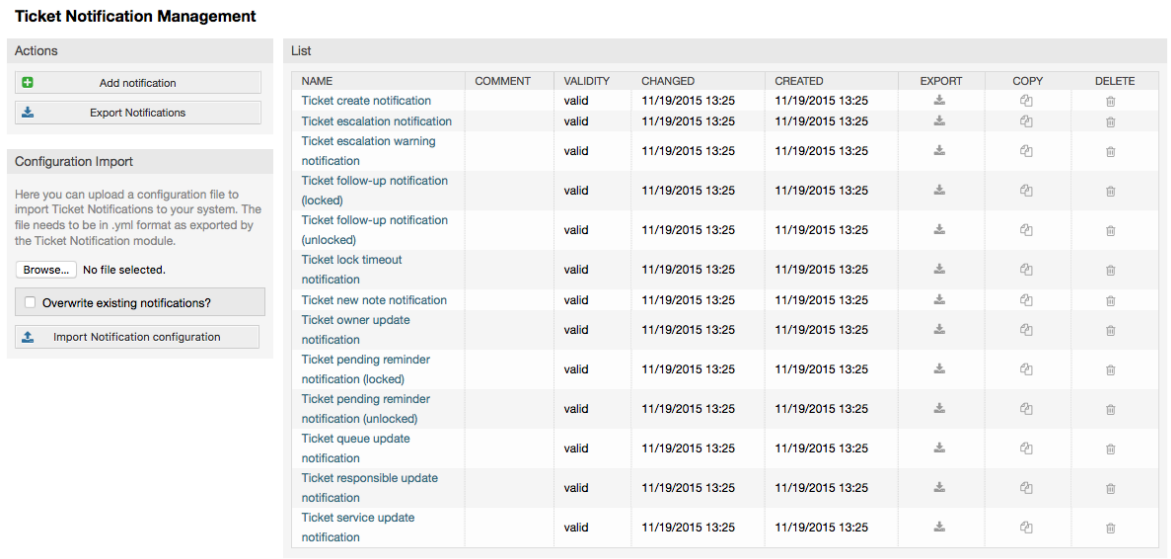
As with other OTRS entities, email addresses cannot be deleted, only deactivated by setting the Valid option to *invalid* or *invalid-temporarily*.

1.8. Ticket Notifications

OTRS allows ticket notifications to be sent to agents and customers, based on the occurrence of certain events. Agents can customize their ticket notification settings via the preferences link.

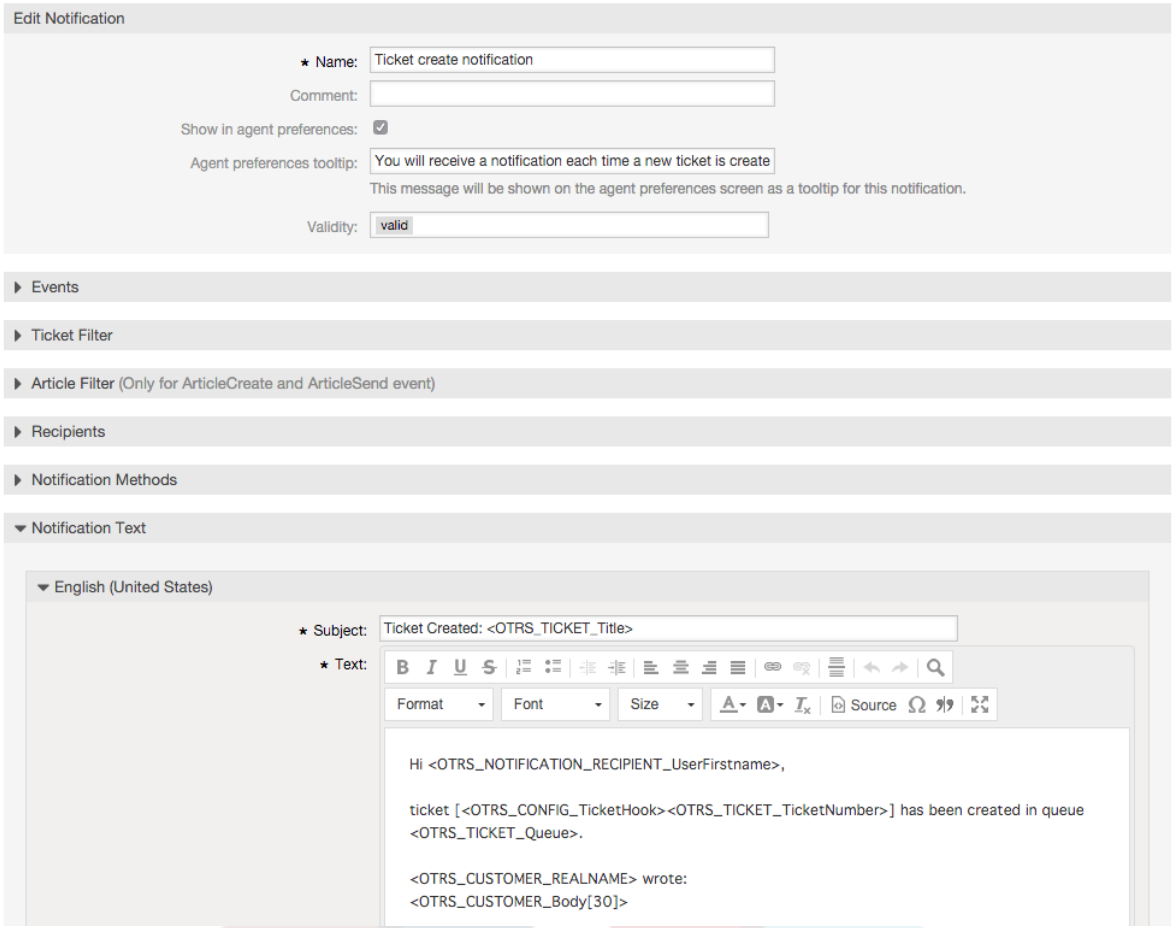
Through the "Ticket Notifications" link on the Admin page, you can manage the ticket notifications of your system (see figure below). OTRS comes with a set of predefined notifications that cover a wide range of use cases.

Figure 4.42. Ticket notification management



You can customize many aspects of the notifications. Click on the notification you want to change, and its content will be loaded for editing (see figure below).

Figure 4.43. Customizing a notification



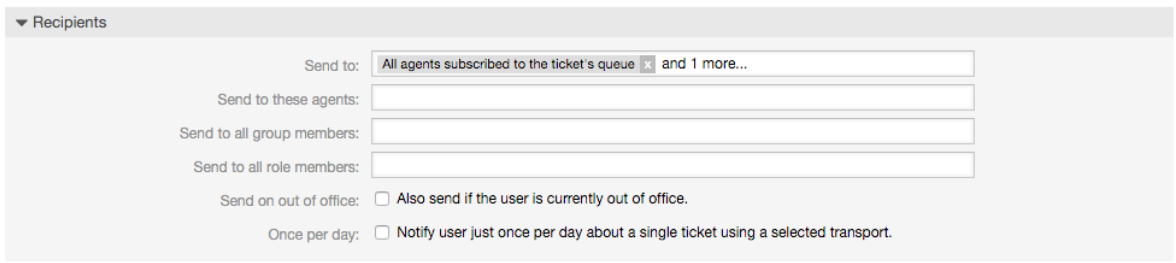
You can edit the basic data of this notification such as name and comment, and control if the agents may choose to receive this notification (per transport method). For every language, a subject and body can be added/edited to configure what will actually be sent as the notification content.

Just as with signatures and salutations, it is possible to dynamically create the content of a notification by using special variables. You can find a list of variables at the bottom of the screen.

You can choose which events should trigger this notification, and limit it to tickets which match certain criteria (ticket and/or article filter). This makes it possible to create different notifications for different queues, priorities or other criteria that might be relevant for your system.

The recipients of the notification can be configured according to different criteria (groups, roles, individual agents etc.). All configured recipients will receive the notification.

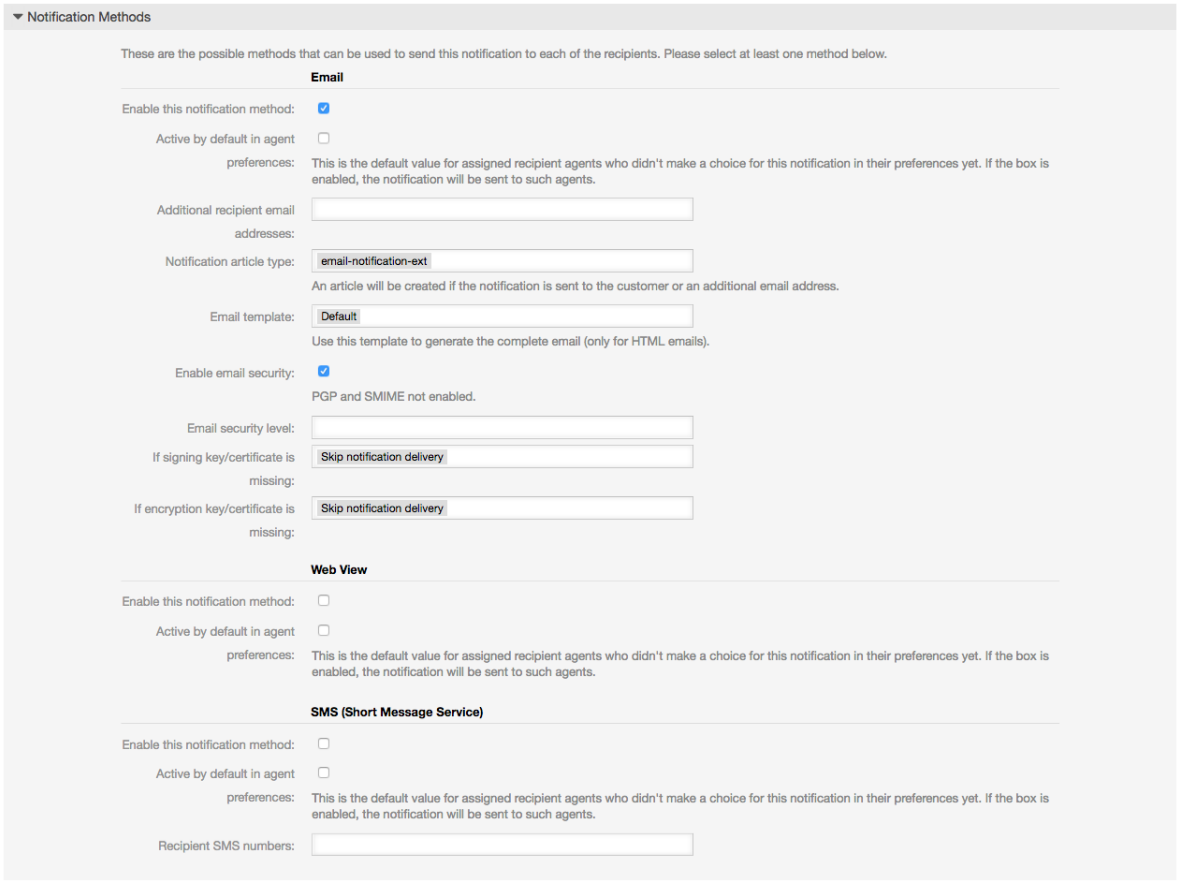
Figure 4.44. Customizing a notification's recipients



Additionally, you can specify if the notification should be sent to agents who are out of office, and limit the sending to once per day and ticket (e. g. pending reminder notification).

Notifications can be sent with different notification methods. The "Email" notification method is available in OTRS Free, with **OTRS Business Solution™** you also get the possibility to store and view the notifications in the database (so that no email client is needed to use OTRS) as well as to send them via SMS (e. g. for very important notifications).

Figure 4.45. Customizing notification methods



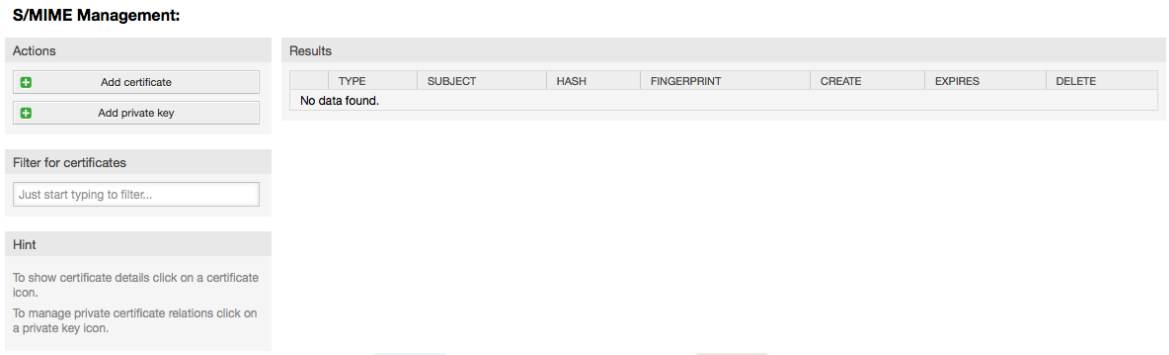
Since OTRS 5S Email transport contains security options for each notification, that includes signing and encrypting possibilities with PGP and SMIME and the opportunity to decide what to do in case of missing key or certificate.

1.9. S/MIME

OTRS can process incoming S/MIME encoded messages and sign outgoing mails. Before this feature can be used, you need to activate it and change some configuration parameters in the SysConfig.

The "S/MIME Certificates" link on the Admin page allows you to manage your S/MIME certificates (see figure below). You can add or remove certificates, and also search through the SMIME data.

Figure 4.46. S/MIME management

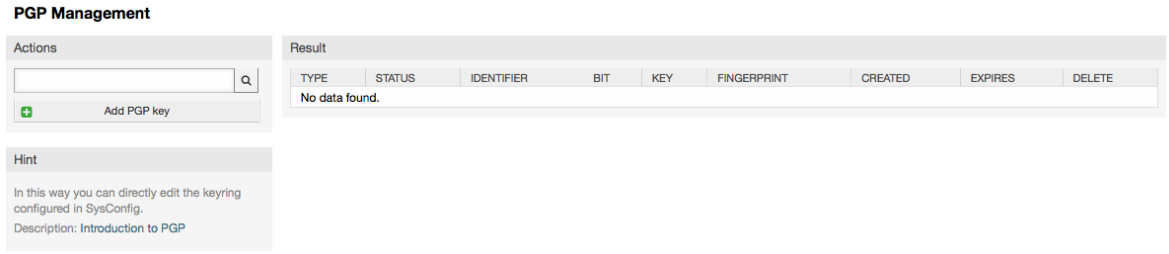


1.10. PGP

OTRS handles PGP keys, which allows you to encrypt/decrypt messages and to sign outgoing messages. Before this feature can be used, you need to activate it and change some configuration parameters in the SysConfig.

Through the "PGP Keys" link on the Admin page, it is possible to manage the key ring of the user who shall be used for PGP with OTRS (see figure below), e.g. the local OTRS user or the web server user. It is possible to add and remove keys and signatures, and you can search through all data in your key ring.

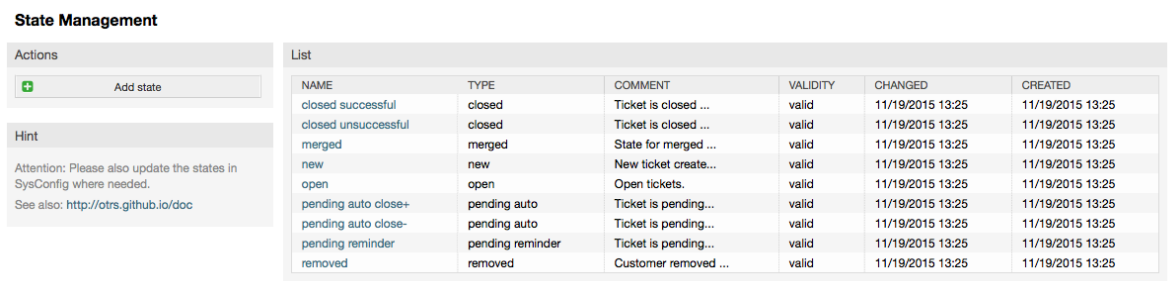
Figure 4.47. PGP management



1.11. States

Through the "States" link on the Admin page, you can manage the different ticket states you want to use in the system (see figure below).

Figure 4.48. State management



After a default setup, there are some states defined:

- closed successful

- closed unsuccessful
- merged
- new
- open
- pending auto close+
- pending auto close-
- pending reminder
- removed

Every state is linked to a type, which needs to be specified if a new state is created. By default the state types are:

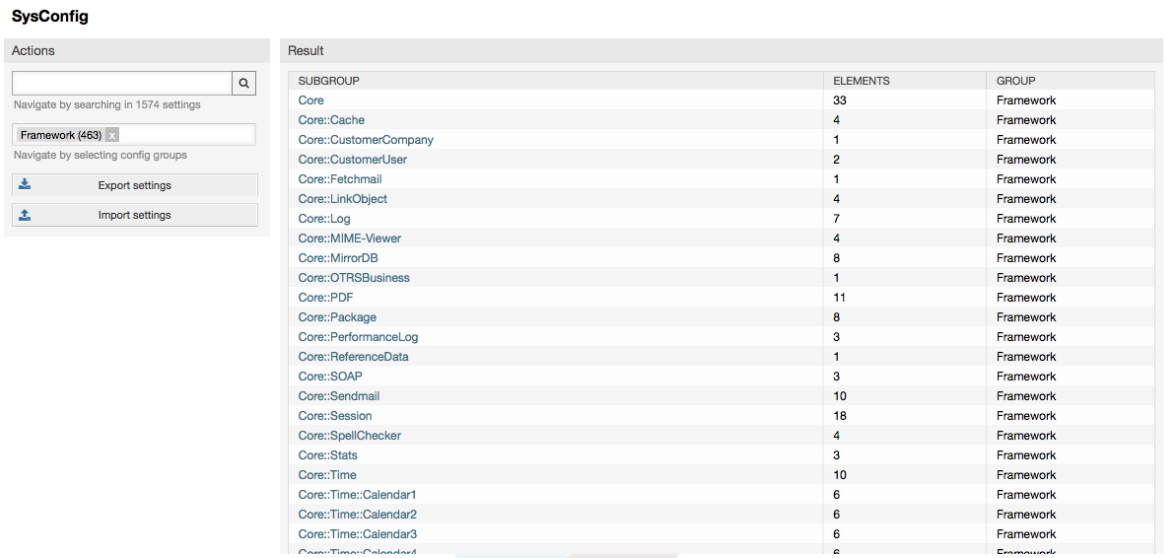
- closed
- merged
- new
- open
- pending auto
- pending reminder
- removed

1.12. SysConfig

The SysConfig link leads to the section where many OTRS configuration options are maintained.

The SysConfig link on the Admin page loads the graphical interface for system configuration (see figure below). You can upload your own configuration files for the system, as well as backup all your current settings into a file. Almost all configuration parameters of the OTRS framework and installed applications can be viewed and changed through this interface. Since all configuration parameters are sorted into groups and sub groups, it is possible to navigate quickly through the vast number of existing parameters. It is also possible to perform a full-text search through all of the configuration parameters.

Figure 4.49. The graphical interface for system configuration (SysConfig)

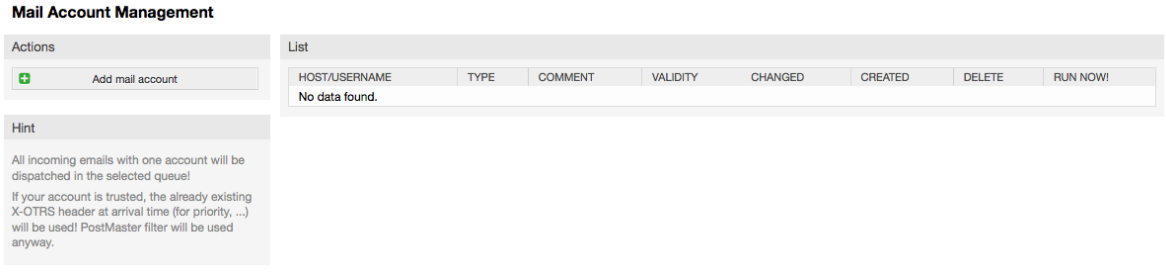


The graphical interface for system configuration is described in more detail in the chapter "Configuring the system through the web interface".

1.13. Using Mail Accounts

There are several possibilities to transport new emails into the ticket system. One way is to use a local MTA and the otrs.PostMaster.pl script that pipes the mails directly into the system. Another possibility is the use of mail accounts which can be administrated through the web interface. The "PostMaster Mail Accounts" link on the Admin page loads the management console for the mail accounts (see figure below). OTRS supports the mail protocols: POP3, POP3S, IMAP and IMAPS.

Figure 4.50. Mail account management

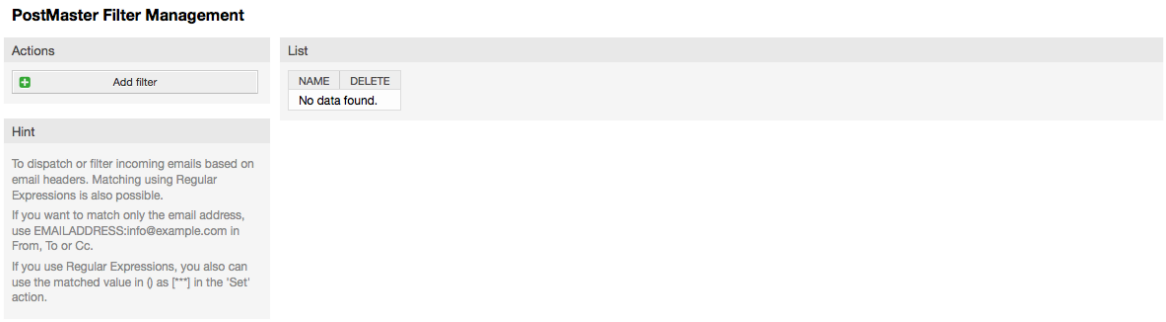


See the section about PostMaster Mail Accounts for more details.

1.14. Filtering Incoming Email Messages

OTRS has the capability to filter incoming email messages. For example, it is possible to put certain emails automatically into specified queues, or to set a specific state or ticket type for some mails. The filters apply to all incoming mails. You can manage your filters via the link "PostMaster Filter" on the Admin page (see figure below).

Figure 4.51. PostMaster filter management



A filter consists of one or more criteria that must be met in order for the defined actions to be executed on the email. Filter criteria may be defined for the headers or the body of an email, e.g. search for specific header entries, such as a sender address, or on strings in the body. Even regular expressions can be used for extended pattern matching. If your filter matches, you can set fields using the X-OTRS headers in the GUI. These values will be applied when creating the ticket or follow-up message in OTRS. The Table 4-5 lists the different X-OTRS headers and their meaning.

Note

You also can use X-OTRS-FollowUp-* headers to set values for follow up emails.

Table 4.5. Function of the different X-OTRS-headers

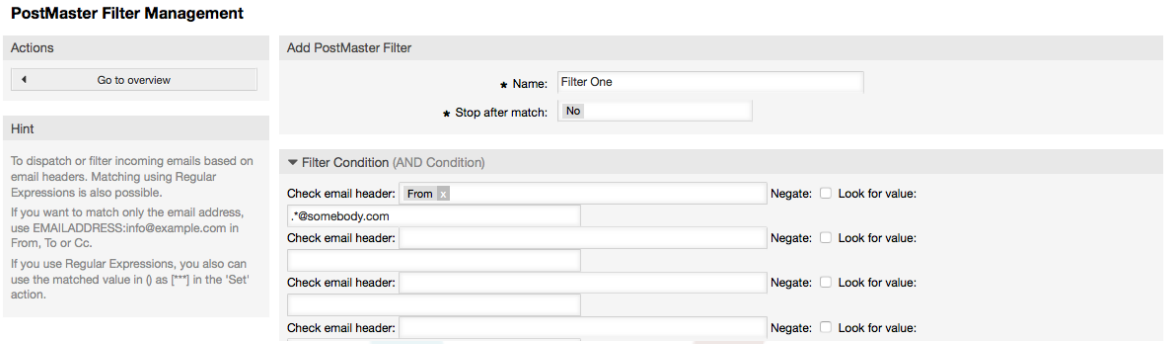
Name	Possible values	Description
X-OTRS-Priority:	1 very low, 2 low, 3 normal, 4 high, 5 very high	Sets the priority of a ticket.
X-OTRS-Queue:	Name of a queue in the system.	Sets the queue where the ticket shall be sorted. If set in X-OTRS header, all other filter rules that try to sort a ticket into a specific queue are ignored. If you use a sub-queue, specify it as "Parent::Sub".
X-OTRS-Lock:	lock, unlock	Sets the lock state of a ticket.
X-OTRS-Ignore:	Yes or True	If this X-OTRS header is set to "Yes", the incoming message will completely be ignored and never delivered to the system.
X-OTRS-State:	new, open, closed successful, closed unsuccessful, ...	Sets the next state of the ticket.
X-OTRS-State-PendingTime:	e. g. 2010-11-20 00:00:00	Sets the pending time of a ticket (you also should sent a pending state via X-OTRS-State). You can specify absolute dates like "2010-11-20 00:00:00" or relative dates, based on the arrival time of the email. Use the form "+ $Number

Name	Possible values	Description
		$Unit", where $Unit can be 's' (seconds), 'm' (minutes), 'h' (hours) or 'd' (days). Only one unit can be specified. Examples of valid settings: "+50s" (pending in 50 seconds), "+30m" (30 minutes), "+12d" (12 days). Note that settings like "+1d 12h" are not possible. You can specify "+36h" instead.
X-OTRS-Type:	default (depends on your setup)	Sets the type of a ticket (if Ticket::Type is activated).
X-OTRS-Service:	(depends on your setup)	Sets the service of a ticket (if Ticket::Service is active). If you want to set a sub-service you should specify it as "Parent::Sub".
X-OTRS-SLA:	(depends on your setup)	Sets the SLA of a ticket (if Ticket::Service support is active).
X-OTRS-CustomerUser:	CustomerUser	Sets the customer user for the ticket.
X-OTRS-CustomerNo:	CustomerNo	Sets the customer ID for this ticket.
X-OTRS-SenderType:	agent, system, customer	Sets the type of the ticket sender.
X-OTRS-ArticleType:	email-external, email-internal, email-notification-ext, email-notification-int, phone, fax, sms, webrequest, note-internal, note-external, note-report	Sets the article type for the incoming ticket.
X-OTRS-DynamicField-<DynamicFieldName>:	Depends on Dynamic Field configuration (Text: Notebook, Date: 2010-11-20 00:00:00, Integer: 1)	Saves an additional info value for the ticket on <DynamicFieldName> Dynamic Field.
X-OTRS-Loop:	True	If this X-OTRS header is set, no auto answer is delivered to the sender of the message (mail loop protection).

You should specify a name for every filter rule. Filter criteria can be specified in the section "Filter Condition". Choose via the listboxes for "Header 1", "Header 2" and so on for the parts of the messages where you would like to search, and specify on the right side the values you wish to filter on. In the section "Set Email Headers", you can choose the actions that are triggered if the filter rules match. You can select for "Header 1", "Header 2" and so on to select the X-OTRS-Header and set the associated values (see figure below).

Filter rules are evaluated in alphabetical order, and are all executed except if the "Stop after match" setting has been set to "Yes" in one of the rules (in this case evaluation of the remaining filters is canceled).

Figure 4.52. Add a PostMaster filter



Example 4.1. Sort spam mails into a specific queue

A useful filter rule would be to let OTRS automatically move mails marked for spam, by using a spam detection tool such as SpamAssassin, into the "Junk" queue. SpamAssassin adds the "X-Spam-Flag" header to every checked mail. When the mail is marked as spam, the Header is set to "Yes". So the filter criteria would be "X-Spam-Flag: Yes". To create a filter rule with this criteria you can insert the name as, for example, "spam-mails". In the section for "Filter Condition", choose "X-Spam-Flag:" for "Header 1" from the listbox. Insert "Yes" as value for this header. Now the filter criteria is specified. To make sure that all spam mails are placed into the "Junk" queue, choose in the section for "Set Email Headers", the "X-OTRS-Queue:" entry for "Header 1". Specify "Junk" as value for this header. Finally add the new filter rule to activate it for new messages in the system.

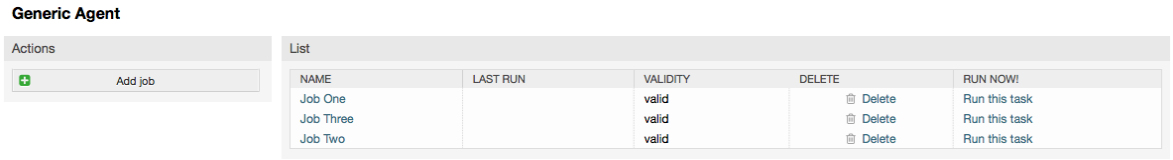
There are additional modules, that can be used to filter incoming messages more specifically. These modules might be useful when dealing with larger, more complex systems.

1.15. Executing Automated Jobs with the GenericAgent

The GenericAgent is a tool to execute tasks automatically. The GenericAgent, for example, can close or move tickets, send notifications on escalated tickets, etc.

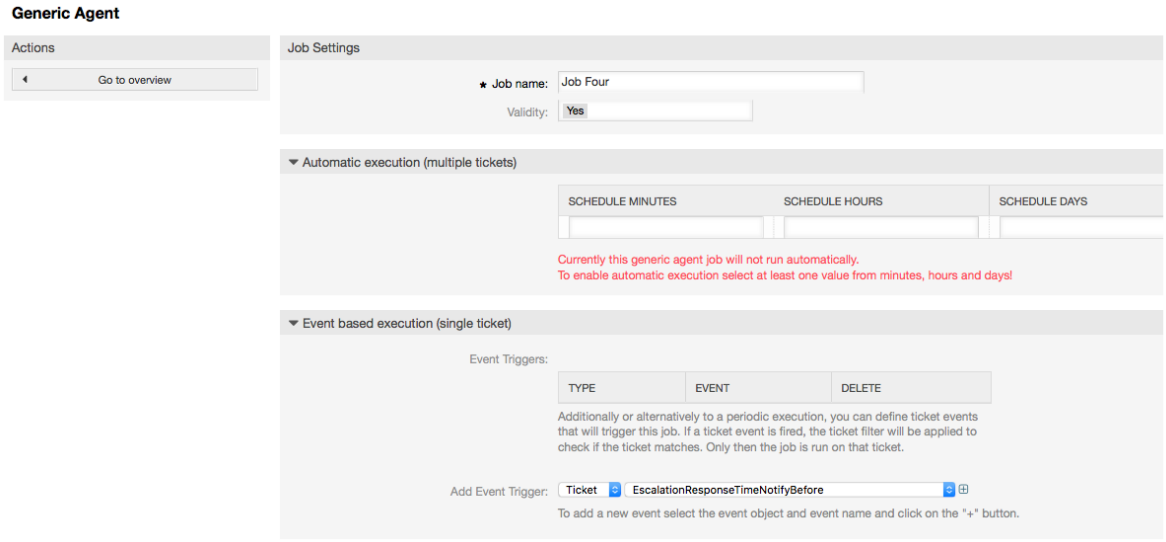
Click the link "GenericAgent" on the Admin page (see figure below). A table with all automated jobs in the system is displayed. These jobs can then be edited, run manually or removed entirely.

Figure 4.53. Job list for the GenericAgent



Click the "Add job" button to create a new job. You first need to supply a name. Then you can specify how the job will be executed: automatic at fixed times (like a cronjob, this mode will operate on all tickets found by the ticket filter) or based on ticket events (right after a single ticket was modified, if it matches the ticket filter). Note that if you manually run event based jobs from the overview screen, they will operate on all tickets fund by the ticket filter.

Figure 4.54. Creating a job for the GenericAgent



For every job, you can specify a ticket filter, for example to only operate on tickets in a certain queue. All filter criteria must be met for a job to be run on a ticket.

Finally, the ticket can be modified by setting various ticket fields like a new queue or state. It is possible to attach a note to the ticket(s) or run a customized module. You also have the option to delete the ticket(s) from the database. This can be useful to purge outdated or invalid data from the system.

Warning

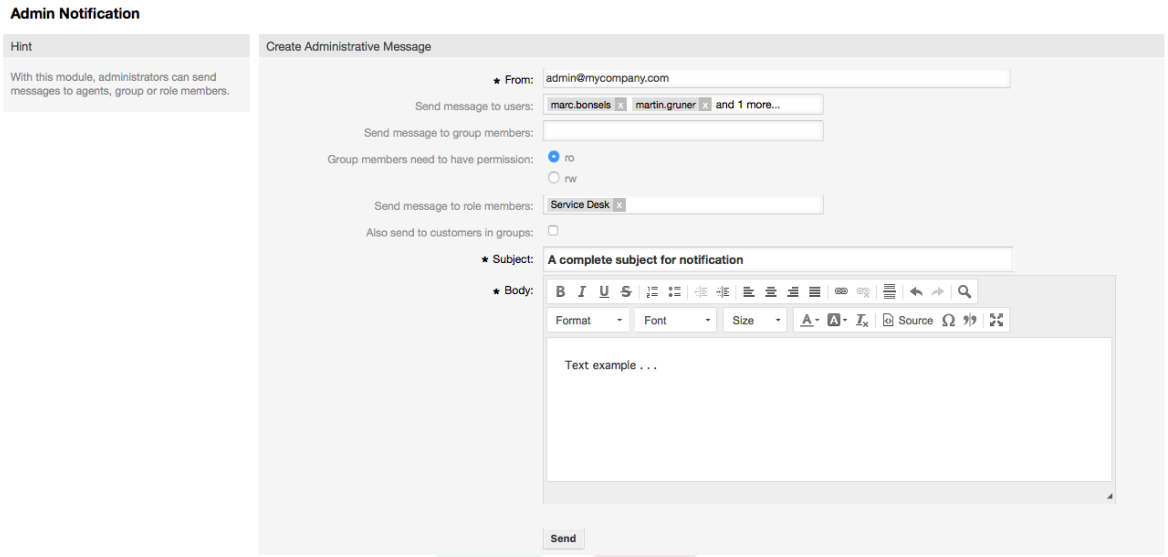
If you use the ticket delete function, all affected tickets and their attachments will be removed from the database and cannot be restored!

After editing a job, OTRS will return to the overview screen. There you have the possibility to run any job manually. If you choose to run a job, you will first see all tickets which will be affected when the job actually is run. This list helps you to verify that the job is working as intended. At this point no changes have been made to these tickets yet. Only if you confirm the screen the job will be executed.

1.16. Administrative Messages

OTRS administrators can send messages to specific users or groups. The "Admin Notification" link on the Admin page opens the screen where the agents and groups that should be notified can be selected (see figure below).

Figure 4.55. Admin notification screen

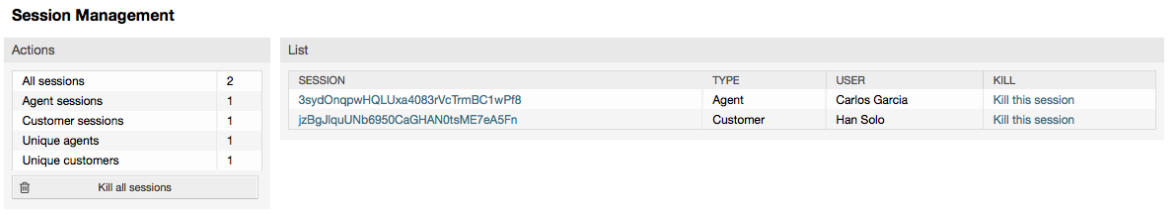


It is possible to specify the sender, subject and body text of the notification. You can also select the agents, groups and roles who should receive the message.

1.17. Session Management

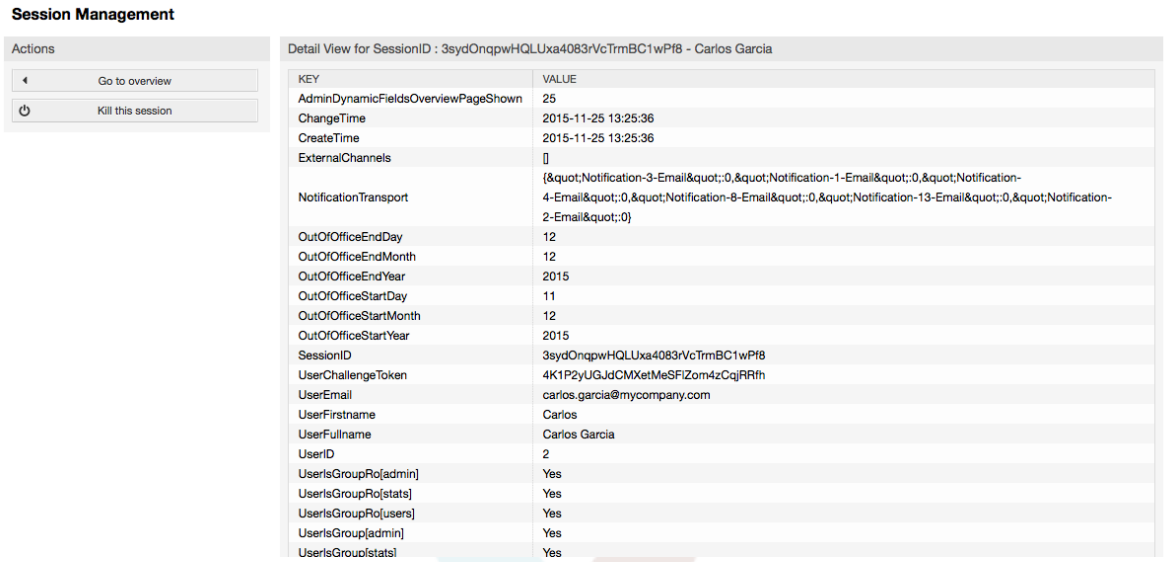
You can see all logged in users and their session details by clicking the "Session Management" link in the admin area (see figure below).

Figure 4.56. Session management



Some statistics about all active sessions are displayed, e.g. how many agents and customer users are logged in and the number of active sessions. Any individual session can be removed by clicking on the *Kill this session* link on the right-hand side of the list. You also have the option to *Kill all sessions*, which can be useful if you want to take the system offline. Detailed information for every session is available, too (see figure below).

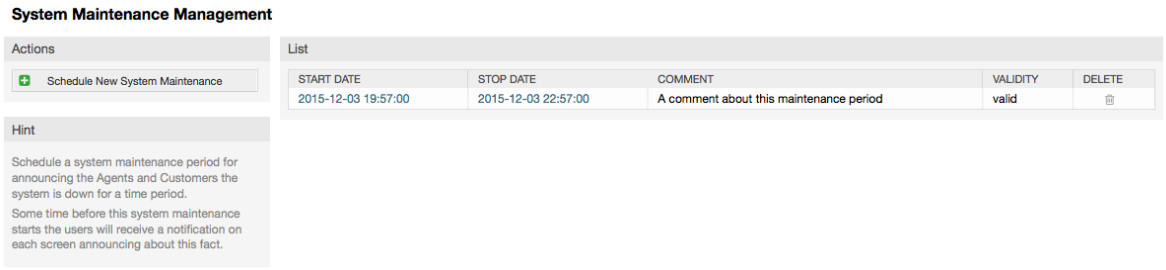
Figure 4.57. Session details



1.18. System Maintenance

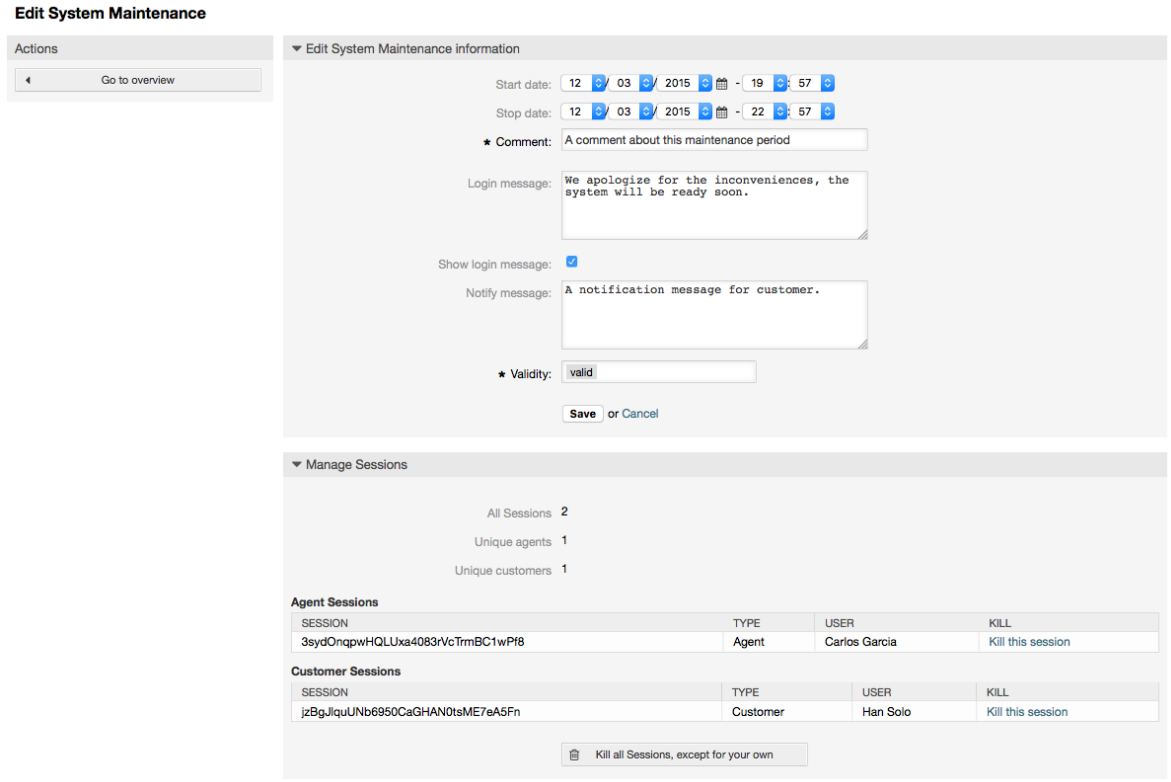
System Maintenance give the option to schedule one or more maintenance periods for the system. During this period no agents or customers can login into the system (except for Agents in the "admin" group). Current logged users and customers receive a notification about the maintenance (before and during the maintenance period). Administrators have the option to kill the sessions for logged agents and customers, all this in preparation to be able to make changes in the system (e.g. a system update) in a "safe" environment.

Figure 4.58. The system maintenance overview screen with some scheduled periods



The Start Date and the Stop Date are required fields, and the only rule for this combination is that Start Date can not be a date after the Stop Date.

Figure 4.59. The system maintenance edit screen

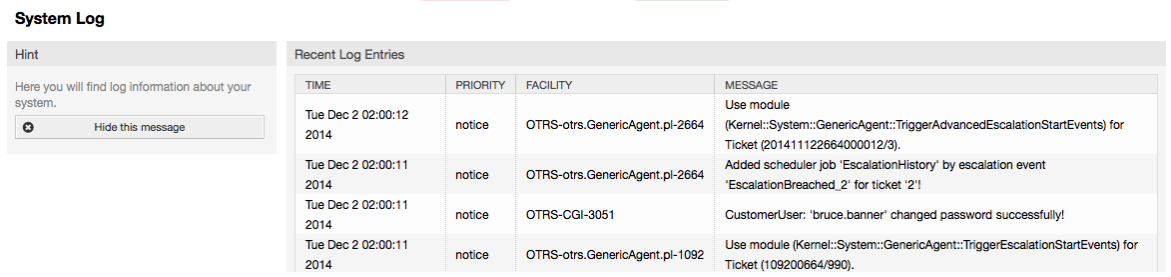


After a new maintenance period is defined an overview and details about the current active sessions is shown, from there administrators can kill this sessions one by one or all of them (except current) if it is needed.

1.19. System Log

The "System Log" link on the Admin page shows the log entries of the system, reverse chronologically sorted with most recent first (see figure below).

Figure 4.60. System Log



Each line in the log contains a time stamp, the log priority, the system component and the log entry itself.

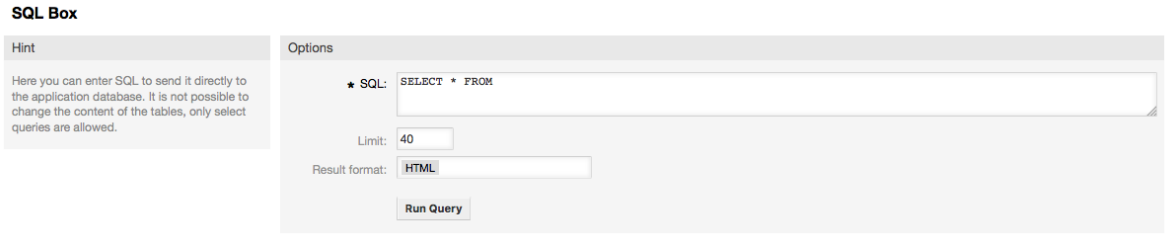
Note

System logs are available via the web interface only on Linux / Unix systems.

1.20. SQL Queries via the SQL Box

The "SQL Box" link on the Admin page opens a screen that lets you query the content of the tables in the OTRS database (see figure below). It is not possible to change the content of the tables, only 'select' queries are allowed.

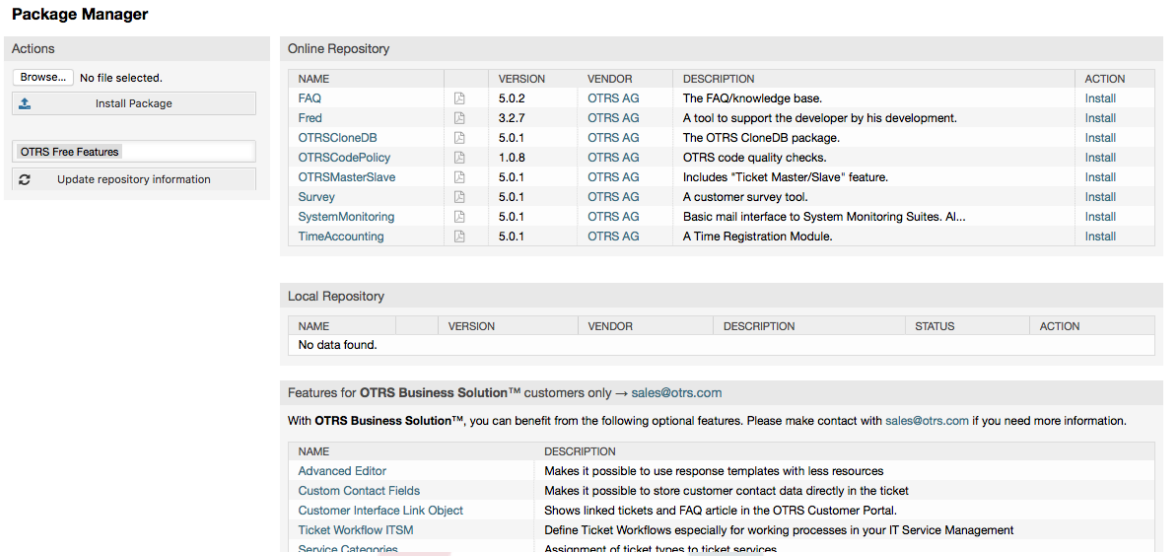
Figure 4.61. SQL Box



1.21. Package Manager

Using the "Package Manager" link on the Admin page, you can install and manage packages that extend the functionality of OTRS (see figure below). See the Additional applications section for a discussion on the extensions that are available from the OTRS repositories.

Figure 4.62. Package Manager



The Package Manager shows the OTRS addon packages you currently have installed on your server, together with their version numbers.

You can install packages from a remote host by selecting the repository in the *Online Repository* section, and clicking the *Update repository information* button. The available packages are displayed in the corresponding table. The right side of the screen shows the available packages. To install a package, click on *Install*. After installation, the package is displayed in the *Local Repository* section.

To upgrade an installed package, the list of available packages in the online repository will show *Upgrade* in the Action column for any package that has a higher version than the one that is installed locally. Just click Upgrade and it will install the new package version on your system.

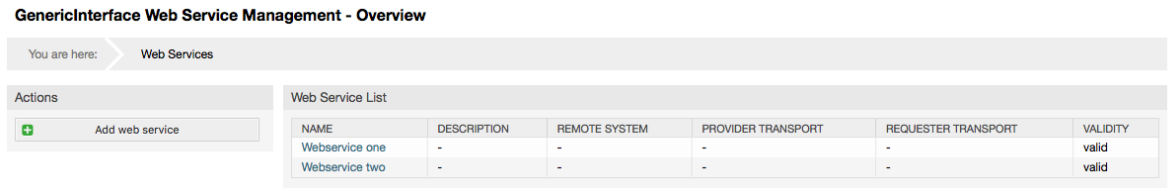
In some cases, such as when your OTRS system is not connected to the Internet, you can also install those packages that you have downloaded to a local disk. Click the *Browse* button on the Actions side bar, and select the .opm file of the package on your disk. Click *Open* and then *Install Package*. After the installation has been completed, the package is displayed in the *Local Repository* section. You can use the same steps for updating a package that is already installed.

In special cases, you might want to configure the Package Manager, e.g., to use a proxy or to use a local repository. Just take a look at the available options in SysConfig under Framework:Core::Package.

1.22. Web Services

The Web Services link leads to the graphical interface where web services (for the OTRS Generic Interface) are created and maintained (see figure below).

Figure 4.63. The graphical interface for web services

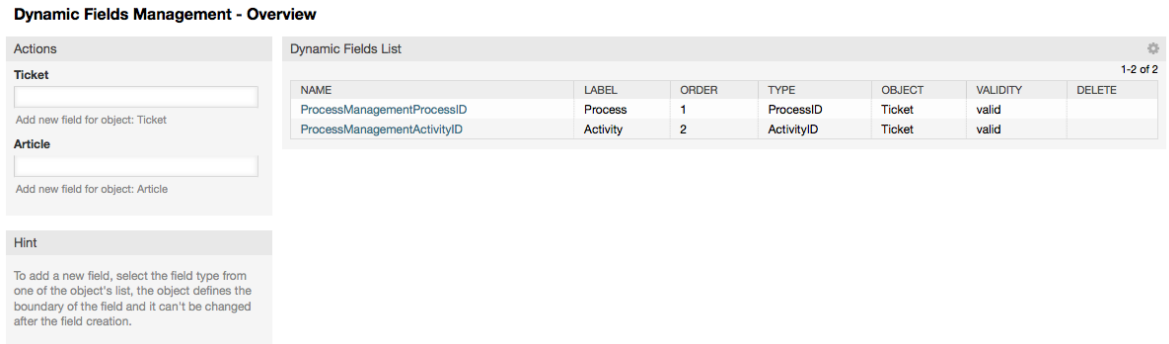


The graphical interface for web services configuration is described in more detail in the section "Web Service Graphical Interface".

1.23. Dynamic Fields

Dynamic Fields is the place where you setup and manage custom fields for tickets and articles (see figure below).

Figure 4.64. The dynamic fields overview screen with some dynamic fields



The dynamic fields configuration is described in more detail in the section "Dynamic Fields Configuration".

Each dynamic field type has its own configuration settings and therefore its own configuration screen.

Note

In the OTRS framework, dynamic fields can only be linked to tickets and articles by default, but they can be extended to other objects as well.

2. System Configuration

2.1. OTRS config files

All OTRS configuration files are stored in the directory `Kernel` and in its subdirectories. There is no need to manually change any other file than `Kernel/Config.pm`, because the rest of the files will be changed when the system gets upgraded. Just copy the configuration parameters from the other files into `Kernel/Config.pm` and change them as per your needs. This file will never be touched during the upgrade process, so your manual settings are safe.

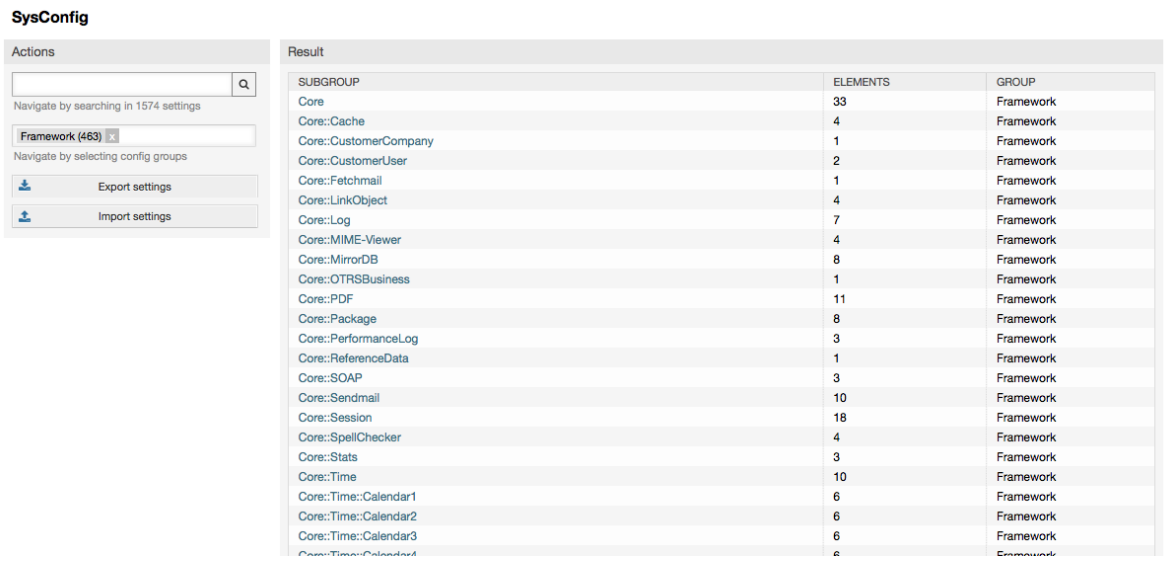
In the directory `Kernel/Config/Files` there are some other files that are parsed when the OTRS login page is accessed. If additional applications like the FAQ or the File Manager are installed, the configuration files for those can also be found in the mentioned path.

If the OTRS web interface is accessed, all .xml files in the `Kernel/Config/Files` directory are parsed in alphabetical order, and the settings for the central framework and additional applications will be loaded. Afterwards, the settings in the files `Kernel/Config/Files/ZZZAAuto.pm`, `Kernel/Config/Files/ZZZAuto.pm` and `Kernel/Config/Files/ZZZProcessManagement.pm` (if it exists) will be evaluated. These files are used by the graphical interface for system configuration caching and should never be changed manually. Lastly, the file `Kernel/Config.pm` that contains your individual settings and manually changed configuration parameters, will be parsed. Reading the configuration files in this order makes sure that your specific configuration settings are used by the system.

2.2. Configuring the System Through the Web Interface

Since OTRS 2.0, nearly all configuration parameters of the central framework or additional installed applications, can be changed easily with the graphical interface for system configuration. Log in as OTRS administrator and follow the SysConfig link on the Admin page to execute the new configuration tool (see figure below).

Figure 4.65. The graphical interface for system configuration



OTRS currently has over 600 configuration parameters, and there are different ways to quickly access a specific one. With the full text search, all configuration parameters can

be scanned for one or more keywords. The full text search not only searches through the names of the configuration parameters, but also through the descriptions of the parameters. This allows an element to be found easily even if its name is unknown.

Furthermore, all configuration parameters are sorted in main groups and sub groups. The main group represents the application that the configuration parameter belongs to, e.g. "Framework" for the central OTRS framework, "Ticket" for the ticket system, "FAQ" for the FAQ system, and so on. The sub groups can be accessed if the application is selected from the groups listbox and the "Select group" button is pressed.

Every configuration parameter can be turned on or off via a checkbox. If the parameter is turned off, the system will ignore this parameter or use a default. It is possible to switch a changed configuration parameter back to the system default using the Reset link. The Update button submits all changes to system configuration parameters.

If you want to save all the changes you made to your system's configuration, for example to setup a new installation quickly, you can use the "Export settings" button, which will create a .pm file. To restore your own settings, just press the "Import settings" and select the .pm created before.

Note

For security reasons, the configuration parameters for the database connection cannot be changed in the SysConfig section. They have to be set manually in `Kernel/Config.pm`.

3. Backing Up the System

This chapter describes the backup and restore of the OTRS data.

3.1. Backup

There are two types of data to backup: application files (e.g. the files in `/opt/otrs`), and the data stored in the database.

To simplify backups, the script `scripts/backup.pl` is included with every OTRS installation. It can be run to backup all important data (see Script below).

```
linux:/opt/otrs# cd scripts/
linux:/opt/otrs/scripts# ./backup.pl --help
backup.pl - backup script
Copyright (C) 2001-2014 OTRS AG, http://otrs.com/
usage: backup.pl -d /data_backup_dir/ [-c gzip|bzip2] [-r 30] [-t fullbackup|nofullbackup|
dbonly]
linux:/opt/otrs/scripts#
```

Script: Getting help about the OTRS backup mechanism.

Execute the command specified in the script below to create a backup:

```
linux:/opt/otrs/scripts# ./backup.pl -d /backup/
Backup /backup//2010-09-07_14-28/Config.tar.gz ... done
Backup /backup//2010-09-07_14-28/Application.tar.gz ... done
Dump MySQL rdbms ... done
Compress SQL-file... done
linux:/opt/otrs/scripts#
```

Script: Creating a backup.

All data was stored in the directory `/backup/2010-09-07_14-28/` (see Script below). Additionally, the data was saved into a .tar.gz file.

```
linux:/opt/otrs/scripts# ls /backup/2010-09-07_14-28/
Application.tar.gz  Config.tar.gz  DatabaseBackup.sql.gz
linux:/opt/otrs/scripts#
```

Script: Checking the backup files.

3.2. Restore

To restore a backup, the saved application data has to be written back into the installation directory, e.g. `/opt/otrs`. Also the database has to be restored.

A script `scripts/restore.pl` (see Script below), which simplifies the restore process, is shipped with every OTRS installation. It supports MySQL and PostgreSQL.

```
linux:/opt/otrs/scripts# ./restore.pl --help
restore.pl - restore script
Copyright (C) 2001-2014 OTRS AG, http://otrs.com/
usage: restore.pl -b /data_backup/<TIME>/ -d /opt/otrs/
linux:/opt/otrs/scripts#
```

Script: Getting help about the restore mechanism.

Data that is stored, for example, in the directory `/backup/2010-09-07_14-28/`, can be restored with the command specified in the script below, assuming the OTRS installation is at `/opt/otrs`.

```
linux:/opt/otrs/scripts# ./restore.pl -b /backup/2010-09-07_14-28 -d /opt/otrs/
Restore /backup/2010-09-07_14-28//Config.tar.gz ...
Restore /backup/2010-09-07_14-28//Application.tar.gz ...
create MySQL
decompresses SQL-file ...
cat SQL-file into MySQL database
compress SQL-file...
linux:/opt/otrs/scripts#
```

Script: Restoring OTRS data.

4. Email Settings

4.1. Sending/Receiving Emails

4.1.1. Sending Emails

4.1.1.1. Via Sendmail (Default)

OTRS can send out emails via Sendmail, Postfix, Qmail or Exim. The default configuration is to use Sendmail and should work out-of-the-box.

You can configure the sendmail settings via the graphical configuration frontend (`Framework::Core::Sendmail`)

4.1.1.2. Via SMTP Server or Smarthost

OTRS can send emails via SMTP (Simple Mail Transfer Protocol / RFC 821) or Secure SMTP.

The SMTP server settings can be configured via the SysConfig (`Framework::Core::Sendmail`). If you don't see SMTPS available as an option, the required Perl modules are missing. In that case, please refer to "Installation of Perl modules required for OTRS" for instructions.

4.1.2. Receiving Emails

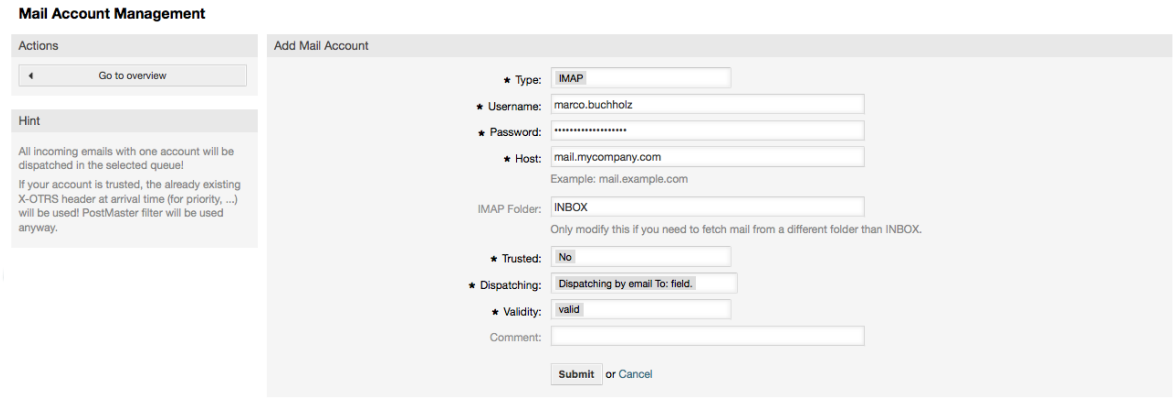
4.1.2.1. Mail Accounts Configured via the OTRS GUI

OTRS is able to receive emails from POP3, POP3S, IMAP and IMAPS mail accounts.

Configure your mail accounts via the "PostMaster Mail Accounts" link on the Admin page.

If a new mail account is to be created (see figure below), then its mail server name, login name and password must be specified. Also, you need to select the mail server type, which can be POP3, POP3S, IMAP or IMAPS. If you don't see your server type available as an option, the required Perl modules are missing on your system. In that case, please refer to "Installation of Perl modules required for OTRS" for instructions.

Figure 4.66. Adding a mail account



If you select Yes for the value of the Trusted option, any X-OTRS headers attached to an incoming message are evaluated and executed. Because the X-OTRS header can execute some actions in the ticket system, you should set the Trusted option to Yes only for known senders. X-OTRS-Headers are used by the filter module in OTRS. The X-OTRS headers are explained in this table in more detail. Any postmaster filter rules created are executed, irrespective of the Trusted option's setting.

The distribution of incoming messages can be controlled if they need to be sorted by queue or by the content of the "To:" field. For the Dispatching field, if "Dispatching by selected queue" is selected, all incoming messages will be sorted into the specified queue. The address where the mail was sent to is disregarded in this case. If "Dispatching by email To: field" is selected, the system checks if a queue is linked with the address in the To: field of the incoming mail. You can link an address to a queue in the E-mail address management section of the Admin page. If the address in the To: field is linked with a queue, the new message will be sorted into the linked queue. If no link is found between the address in the To: field and any queue, then the message flows into the "Raw" queue in the system, which is the PostmasterDefaultQueue after a default installation.

All data for the mail accounts are saved in the OTRS database. The `bin/otrs.Console.pl Maint::PostMaster::MailAccountFetch` command uses the settings in the database and fetches the mail. You can execute it manually to check if all your mail settings are working properly.

On a normal installation, the mail will be fetched every 10 minutes by the OTRS Daemon.

Note

When fetching mail, OTRS deletes the mail from the POP or IMAP server. There is no option to also keep a copy on the server. If you want to retain a copy on the

server, you should create forwarding rules on your mail server. Please consult your mail server documentation for details.

4.1.2.2. Via Command Line Program and Procmail (`otrs.Console.pl Maint::PostMaster::Read`)

If you cannot use mail accounts to get the email into OTRS, the command line program `bin/otrs.Console.pl Maint::PostMaster::Read` might be a way around the problem. It takes the mails via STDIN and pipes them directly into OTRS. That means email will be available in your OTRS system if the MDA (mail delivery agent, e.g. procmail) executes this program.

To test `bin/otrs.Console.pl Maint::PostMaster::Read` without an MDA, execute the command of the following script.

```
linux:/opt/otrs# cd bin
linux:/opt/otrs/bin# cat ../doc/sample_mails/test-email-1.box | ./otrs.Console.pl
 Maint::PostMaster::Read
linux:/opt/otrs/bin#
```

Script: Testing PostMaster without the MDA.

If the email is shown in the QueueView, then your setup is working.

Example 4.2. Routing via Procmail Using `otrs.Console.pl`

In order to route mails in a specific queue using `otrs.Console.pl` use the following example.

```
| $SYS_HOME/bin/otrs.Console.pl Maint::PostMaster::Read --target-queue=QUEUENAME
```

When sorting to a subqueue, you must separate the parent and child queue with a `::`.

```
| $SYS_HOME/bin/otrs.Console.pl Maint::PostMaster::Read --target-queue=QUEUENAME::SUBQUEUE
```

Procmail is a very common e-mail filter in Linux environments. It is installed on most systems. If not, have a look at the *procmail homepage*.

To configure procmail for OTRS (based upon a procmail configured MTA such as sendmail, postfix, exim or qmail), use the `~otrs/.procmailrc.dist` file and copy it to `.procmailrc` and add the lines of the script below.

```
SYS_HOME=$HOME
PATH=/bin:/usr/bin:/usr/local/bin
# --
# Pipe all email into the PostMaster process.
# --
:0 :
| $SYS_HOME/bin/otrs.Console.pl Maint::PostMaster::Read
```

Script: Configuring procmail for OTRS.

All email sent to the local OTRS user will be piped into `bin/otrs.Console.pl Maint::PostMaster::Read` and then shown in your QueueView.

4.1.2.3. Fetching emails via POP3 or IMAP and fetchmail for `otrs.Console.pl Maint::PostMaster::Read`

In order to get email from your mail server, via a POP3 or IMAP mailbox, to the OTRS machine/local OTRS account and to procmail, use fetchmail.

Note

A working SMTP configuration on the OTRS machine is required.

You can use the `.fetchmailrc.dist` in the home directory of OTRS and copy it to `.fetchmailrc`. Modfiy/change it for your needs (see the Example below).

Example 4.3. `.fetchmailrc`

```
#poll (mailserver) protocol POP3 user (user) password (password) is (localuser)
poll mail.example.com protocol POP3 user joe password mama is otrs
```

Don't forget to set the `.fetchmailrc` to 710 (**chmod 710 .fetchmailrc**)!

With the `.fetchmailrc` from the Example above, all email will be forwarded to the local OTRS account, if the command **fetchmail -a** is executed. Set up a cronjob with this command if you want to fetch the mails regularly.

4.1.2.4. Filtering/Dispatching by OTRS/PostMaster Modules (for More Complex Dispatching)

If you use the `bin/otrs.Console.pl Maint::PostMaster::Read` or `bin/otrs.Console.pl Maint::PostMaster::MailAccountFetch` method, you can insert or modify X-OTRS header entries with the PostMaster filter modules. With the X-OTRS headers, the ticket system can execute some actions on incoming mails, sort them into a specific queue, change the priority or change the customer ID, for example. More information about the X-OTRS headers are available in the section about adding mail accounts from the OTRS Admin page.

There are some default filter modules:

Note

The job name (e.g. $Self->{'PostMaster::PreFilterModule'}->{'JobName'}) needs to be unique!

`Kernel::System::PostMaster::Filter::Match` is a default module to match on some email header (e.g. From, To, Subject, ...). It can set new email headers (e.g. X-OTRS-Ignore: yes or X-OTRS-Queue: spam) if a filter rule matches. The jobs of the Example below can be inserted in `Kernel/Config.pm`

Example 4.4. Example jobs for the filter module `Kernel::System::PostMaster::Filter::Match`

```
# Job Name: 1-Match
# (block/ignore all spam email with From: noreply@)
$Self->{'PostMaster::PreFilterModule'}->{'1-Match'} = {
    Module => 'Kernel::System::PostMaster::Filter::Match',
    Match => {
        From => 'noreply@',
    },
    Set => {
        'X-OTRS-Ignore' => 'yes',
    },
};
# Job Name: 2-Match
# (sort emails with From: sales@example.com and Subject: **ORDER**
# into queue 'Order')
$Self->{'PostMaster::PreFilterModule'}->{'2-Match'} = {
    Module => 'Kernel::System::PostMaster::Filter::Match',
    Match => {
```

```
            To => 'sales@example.com',
            Subject => '**ORDER**',
        },
        Set => {
            'X-OTRS-Queue' => 'Order',
        },
    };
```

Kernel::System::PostMaster::Filter::CMD is a default module to pipe the email into an external command. The output is given to STDOUT and if the result is true, then set new email header (e.g. X-OTRS-Ignore: yes or X-OTRS-Queue: spam). The Example below can be used in Kernel/Config.pm

Example 4.5. Example job for the filter module Kernel::System::PostMaster::Filter::CMD

```
    # Job Name: 5-SpamAssassin
    # (SpamAssassin example setup, ignore spam emails)
    $Self->{'PostMaster::PreFilterModule'}->{'5-SpamAssassin'} = {
        Module => 'Kernel::System::PostMaster::Filter::CMD',
        CMD => '/usr/bin/spamassassin | grep -i "X-Spam-Status: yes"',
        Set => {
            'X-OTRS-Ignore' => 'yes',
        },
    };
```

Kernel::System::PostMaster::Filter::ExternalTicketNumberRecognition is a default module that adds the possibility to parse external identifiers, in the email subject, the body or both using regular expressions. It then stores this value in a defined dynamic field. When an email comes in, OTRS will first search for an external identifier and when it finds one, query OTRS on the pre-defined dynamic field. If it finds an existing ticket, it will update this ticket, otherwise it will create a new ticket with the external reference number in the separate field.

OTRS SysConfig already provide 4 different settings to setup different external ticket numbers. If more settings are needed they need to be added manually. The following example can be used in Kernel/Config.pm to extend SysConfig settings.

Example 4.6. Example job for the filter module Kernel::System::PostMaster::Filter::ExternalTicketNumberRecognition

```
    # Job Name: ExternalTicketNumberRecognition
    # External Ticket Number Reconition, check for Incident-<number> in incoming mails
subject and
    # body from the addeesses <sender>@externalticket.com, if number is found it will be
stored in
    # the dynamic field 'ExternalNumber' (that need to be setup in the Admin Panel).
    $Self->{'PostMaster::PreFilterModule'}->{'000-ExternalTicketNumberRecognition'} = {
        'FromAddressRegExp' => '\\s*@externalticket.com',
        'NumberRegExp'      => 'Incident-(\\d.*)',
        'SearchInSubject'   => '1',
        'SearchInBody'      => '1',
        'TicketStateTypes'  => 'new;open'
        'DynamicFieldName'  => 'ExternalNumber',
        'Module'            =>
'Kernel::System::PostMaster::Filter::ExternalTicketNumberRecognition',
        'Name'              => 'Test External Ticket Number',
        'SenderType'        => 'system',
    };
```

Configuration Options

- FromAddressRegExp

This is an optional setting. Only mails matching this "From:" address will be considered for this filter. You can adjust this setting to the sender address your external system uses for outgoing mails. In case this address can differ, you can set this option to empty. OTRS will in that case not check the sender address.

- NumberRegExp

  This is a mandatory setting. This setting contains the regular expression OTRS will use to extract the ticket number out of the subject and/or ticket body. The default regular expression will match occurrences of for example 'Incident-12354' and will put the part between parentheses in the dynamic field field, in this case '12354'.

- SearchInSubject

  If this is set to '1', the email subject is searched for a ticket number.

- SearchInBody

  If this is set to '1', the email body is searched for a ticket number.

- TicketStateTypes

  This is an optional setting. If given, it will search OTRS only for open external tickets of given state types. The state types are separated with semicolons.

- DynamicField

  This is a required setting. It defines the dynamic filed that is used to store the external number (the field name must exist in the system and has to be valid).

- SenderType

  This defines the sender type used for the articles created in OTRS.

`Kernel::System::PostMaster::Filter::Decrypt` is a default module that is capable to decrypt an encrypted incoming email message (S/MIME or PGP) placing the unencrypted message body in the email header X-OTRS-BodyDecrypted to be processed later. Additionally it can also update the email body to the unencrypted version.

In order to decrypt the emails that system needs to be properly configured for S/MIME and or PGP and have the needed private keys to decrypt the information.

This module is disabled by default and it can be configured directly in the System Configuration in the Admin Panel

Configuration Options

- StoreDecryptedBody

  Set this option to "1" to update the email body to the unencrypted version if the decryption was successful. Be aware that using this the emails will be stored unencrypted and there is no possible way to revert this action.

Of course it's also possible to develop your own PostMaster filter modules.

4.1.2.5. Troubleshooting Email Filtering

This section shows some common issues and things to consider when troubleshooting Postmaster filters.

- The filters are worked in order of their alphabetically sorted filter names. The last filter wins for a certain field to be set, when the criteria match twice.

- "Stop After Match" can prevent a second match.
- Make sure the regular expression is valid.
- Headers can be set as to control OTRS, but are not written in the mail itself.
- When matching one From, CC, TO, use EMAILADDRESS: <your@address>
- The Mailbox must be trusted.
- The match criteria are AND conditions.
- Ticket properties can not be matched by the postmaster filter.

4.2. Secure Email with PGP

OTRS has the capability to sign or encrypt outgoing messages with PGP. Furthermore, encrypted incoming messages can be decrypted. Encryption and decryption are done with the GPL tool GnuPG. To setup GnuPG for OTRS, the following steps have to be performed:

1. Install GnuPG, via the package manager of your operating system.
2. Configure GnuPG for use with OTRS. The necessary directories for GnuPG and a private key have to be created. The command shown in the script below has to be executed as user 'otrs' from a shell.

```
linux:~# su otrs
linux:/root$ cd
linux:~$ pwd
/opt/otrs
linux:~$ gpg --gen-key
gpg (GnuPG) 1.4.2; Copyright (C) 2005 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

gpg: directory `/opt/otrs/.gnupg' created
gpg: new configuration file `/opt/otrs/.gnupg/gpg.conf' created
gpg: WARNING: options in `/opt/otrs/.gnupg/gpg.conf' are not yet active during t
his run
gpg: keyring `/opt/otrs/.gnupg/secring.gpg' created
gpg: keyring `/opt/otrs/.gnupg/pubring.gpg' created
Please select what kind of key you want:
   (1) DSA and Elgamal (default)
   (2) DSA (sign only)
   (5) RSA (sign only)
Your selection? 1
DSA keypair will have 1024 bits.
ELG-E keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
Requested keysize is 2048 bits
Please specify how long the key should be valid.
     0 = key does not expire
  <n>  = key expires in n days
  <n>w = key expires in n weeks
  <n>m = key expires in n months
  <n>y = key expires in n years
Key is valid for? (0)
Key does not expire at all
Is this correct? (y/N) y

You need a user ID to identify your key; the software constructs the user ID
from the Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: Ticket System
Email address: support@example.com
```

```
Comment: Private PGP Key for the ticket system with address support@example.com
You selected this USER-ID:
"Ticket System (Private PGP Key for the ticket system with address support@examp
le.com) <support@example.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
You need a Passphrase to protect your secret key.

Passphrase: secret
Repeat passphrase: secret

We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
++++++++++.++++++++++++++++++++++++....+++++.+++++...++++++++++++++++++++++++.
++++++++++++++++++++++++.+++++.+++++.++++++++++++++++++++++++>++++++++++>+++++
.......>+++++<+++++.............................+++++

Not enough random bytes available.  Please do some other work to give
the OS a chance to collect more entropy! (Need 280 more bytes)

++++++++++.+++++..++++++++++..+++++....++++++++++++++++++++.+++++++++++++++.++++
++++++++++++++++++++++++++.++++++++++.+++++++++++++++.++++++++++.+++++++++++++++
..+++++>.+++++....>+++++.......................................................
.......................................................>+++++<+++++.........
............+++++^^^
gpg: /opt/otrs/.gnupg/trustdb.gpg: trustdb created
gpg: key 7245A970 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
pub   1024D/7245A970 2006-02-03
      Key fingerprint = 2ED5 BC36 D2B6 B055 7EE1  5833 1D7B F967 7245 A970
 uid                  Ticket System (Private pgp key for ticket system with addre
ss support@example.com) <support@example.com>
sub   2048g/52B97069 2006-02-03

linux:~$
```

Script: Configuring GnuPG.

As shown in the script below, the default settings can be applied for most of the required parameters. Only the values for the key owner have to be entered correctly, with a proper password specified for the key.

3. In the screen for the PGP settings, PGP should be activated for OTRS (first option). Also, the path to the gpg program should be set and checked.

   The next config setting (PGP::Options) may also require changing. Via this config setting, the parameters that are used for every execution of gpg by the 'otrs' user can be specified. In particular, the directory of the config files for GnuPG of the 'otrs' user is important. In the example `/opt/otrs/.gnupg` is used. This directory was created earlier during the PGP configuration.

   Via the next config option (PGP::Key::Password) it is possible to specify the pairs of key IDs and their passwords for own private keys. Because communication partners from outside write to the ticket system with their messages encrypted with your public key, OTRS can decrypt these messages with the ID/passwords specified here.

   How to get the id of your own private key? The ID of your own private key is already shown during the key generation (see step 1 from above). It is also possible to get the ID if the command specified in the following script is executed as user 'otrs':

```
```

```
linux:~# su otrs
linux:/root$ cd
linux:~$ pwd
/opt/otrs
linux:~$ gpg --list-keys
/opt/otrs/.gnupg/pubring.gpg
----------------------------
pub   1024D/7245A970 2006-02-03
uid                  Ticket System (Private pgp key for ticket system with
address support@example.com) <support@example.com>
sub   2048g/52B97069 2006-02-03

linux:~$
```

Script: Getting the ID of your own private key.

The ID of the private key can be found in the line that starts with "sub". It is a hexadecimal string that is eight characters long, in the example above it is "52B97069". The password you have to specify for this key in the ticket system is the same that was given during key generation.

After this data is inserted, the "Update" button can be used to save the settings. OTRS is ready to receive and decrypt encoded messages now.

4. Finally, import a customer's public key. This ensures that encrypted messages can be sent out to this customer. There are two ways to import a public key of a customer.

   The first way is to specify the public key of a customer in the customer management interface.

   The second way is to specify the key via the PGP settings, reachable from the Admin page. On the right section of this screen, all already imported public keys of customers are displayed. After PGP has been activated and configured for OTRS, your own public key should also be listed there. In the left area of the PGP setting screen it is possible to search for keys. Also, a new public key can be uploaded into the system from a file.

   The files with the public key that need to be imported into OTRS have to be GnuPGP compatible key files. In most cases, the key stored in a file is an "ASCII armored key". OTRS can deal with this format.

4.3. Secure Email with S/MIME

At first glance, encryption with S/MIME seems a little more complicated than with PGP. First, you have to establish a Certification Authority (CA) for the OTRS system. The subsequent steps are very much like those needed with PGP: configure OTRS, install your own certificate, import other public certificates as needed, etc.

The S/MIME configuration is conducted outside the OTRS web interface for the most part, and should be carried out in a shell by the 'otrs' user. The MIME configuration under Linux is based on SSL (OpenSSL). Therefore, check first of all whether the OpenSSL package is installed on your system. The OpenSSL package includes a script called CA.pl, with which the most important steps of certificate creation can be performed. To simplify the procedure, find out where in the filesystem the CA.pl script is stored and enter the location temporarily into the PATH variable of the shell (see Script below).

```
otrs@linux:~> rpm -ql openssl | grep CA
/usr/share/ssl/misc/CA.pl
otrs@linux:~> export PATH=$PATH:/usr/share/ssl/misc
otrs@linux:~> which CA.pl
/usr/share/ssl/misc/CA.pl
otrs@linux:~> mkdir tmp; cd tmp
otrs@linux:~/tmp>
```

Script: Configuring S/MIME.

The script above shows that a new temporary directory ~/tmp has been created, in which the certificate is to be generated.

To create a certificate, perform the following operations in the command line (we assume that the OTRS administrator has to create a SSL certificate for test and learning purposes. In case you already have a certified SSL certificate for the encryption, use it and skip these steps):

1. Establish your own Certification Authority for SSL. You need it to certify the request for your own SSL certificate (see Script below).

```
otrs@linux:~/tmp> CA.pl -newca
CA certificate filename (or enter to create)

Making CA certificate ...
Generating a 1024 bit RSA private key
...++++++
......++++++
writing new private key to './demoCA/private/cakey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:OTRS-state
Locality Name (eg, city) []:OTRS-town
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Your company
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:OTRS Admin
Email Address []:otrs@your-domain.tld
otrs@linux:~/tmp> ls -la demoCA/
total 8
-rw-r--r--  1 otrs otrs 1330 2006-01-08 17:54 cacert.pem
drwxr-xr-x  2 otrs otrs   48 2006-01-08 17:53 certs
drwxr-xr-x  2 otrs otrs   48 2006-01-08 17:53 crl
-rw-r--r--  1 otrs otrs    0 2006-01-08 17:53 index.txt
drwxr-xr-x  2 otrs otrs   48 2006-01-08 17:53 newcerts
drwxr-xr-x  2 otrs otrs   80 2006-01-08 17:54 private
-rw-r--r--  1 otrs otrs   17 2006-01-08 17:54 serial
otrs@linux:~/tmp>
```

Script: Establishing a Certification Authority for SSL.

2. Generate a certificate request (see Script below).

```
otrs@linux:~/tmp> CA.pl -newreq
Generating a 1024 bit RSA private key
.........................................++++++
....++++++
writing new private key to 'newreq.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
```

```
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:DE\keyreturn
State or Province Name (full name) [Some-State]:OTRS-state
Locality Name (eg, city) []:OTRS-town
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Your company
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:OTRS admin
Email Address []:otrs@your-domain.tld

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Request (and private key) is in newreq.pem
otrs@linux:~/tmp> ls -la
total 4
drwxr-xr-x  6 otrs otrs  232 2006-01-08 17:54 demoCA
-rw-r--r--  1 otrs otrs 1708 2006-01-08 18:04 newreq.pem
otrs@linux:~/tmp>
```

Script: Creating a certificate request.

3. Signing of the certificate request. The certificate request can either be signed and thereby certified by your own CA, or made more credible by being signed by another external certified CA (see Script below).

```
otrs@linux:~/tmp> CA.pl -signreq
Using configuration from /etc/ssl/openssl.cnf
Enter pass phrase for ./demoCA/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number:
            fd:85:f6:9f:14:07:16:c8
        Validity
            Not Before: Jan  8 17:04:37 2006 GMT
            Not After : Jan  8 17:04:37 2007 GMT
        Subject:
            countryName               = DE
            stateOrProvinceName       = OTRS-state
            localityName              = OTRS-town
            organizationName          = Your Company
            commonName                = OTRS administrator
            emailAddress              = otrs@your-domain.tld
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                01:D9:1E:58:C0:6D:BF:27:ED:37:34:14:D6:04:AC:C4:64:98:7A:22
            X509v3 Authority Key Identifier:
                keyid:10:4D:8D:4C:93:FD:2C:AA:9A:B3:26:80:6B:F5:D5:31:E2:8E:DB:A8
                DirName:/C=DE/ST=OTRS-state/L=OTRS-town/O=Your Company/
                CN=OTRS admin/emailAddress=otrs@your-domain.tld
                serial:FD:85:F6:9F:14:07:16:C7

Certificate is to be certified until Jan  8 17:04:37 2007 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
Signed certificate is in newcert.pem
otrs@linux:~/tmp>
```

Script: Signing of the certificate request.

4. Generate your own certificate, and all data going with it, using the signed certificate request (see Script below).

```
otrs@linux:~/tmp> CA.pl -pkcs12 "OTRS Certificate"
Enter pass phrase for newreq.pem:
Enter Export Password:
Verifying - Enter Export Password:
otrs@linux:~/tmp> ls -la
total 12
drwxr-xr-x  6 otrs otrs  328 2006-01-08 18:04 demoCA
-rw-r--r--  1 otrs otrs 3090 2006-01-08 18:13 newcert.p12
-rw-r--r--  1 otrs otrs 3791 2006-01-08 18:04 newcert.pem
-rw-r--r--  1 otrs otrs 1708 2006-01-08 18:04 newreq.pem
otrs@linux:~/tmp>
```

Script: Generating a new certificate.

Now that these operations have been performed, the S/MIME setup must be completed in OTRS.

This part of the setup is carried out in the Admin page, choosing the link "SMIME". In case the general S/MIME support in OTRS has not yet been enabled, the mask points this out to the administrator and provides an appropriate link for enabling it.

With the SysConfig group "Crypt::SMIME", you can also enable and configure the general S/MIME support.

Here you can activate S/MIME support, and define the paths for the OpenSSL command and the directory for the certificates. The key file created above must be stored in the directory indicated here. Otherwise OpenSSL cannot use it.

The next step is performed in the S/MIME configuration on the OTRS Admin page. Here, you can import the private key(s) of the OTRS system and the public keys of other communication partners. Enter the public key that has been created in the beginning of this section and added to OTRS.

Obviously, all public S/MIME keys of communication partners can be imported using the customer administration tool as well.

4.3.1. Fetch S/MIME Certificates from Customer User Backends

It is possible to use a Customer User Backed (such as LDAP) as the source of public S/MIME certificates, this certificates could be imported into the system and be displayed in S/MIME configuration on the OTRS Admin page and they can be used from OTRS to send encrypted emails to the customers.

In order to enable this feature is needed to:

1. Enable 'SMIME' in SysConfig

2. Enable 'SMIME::FetchFromCustomer' in sysConfig

3. Configure a customer user backend to provide the attribute 'UserSMIMECertificate' with the customer user S/MIME certificate (there is an example for LDAP customer user mapping in $OTRS_HOME/Kernel/Config/Defaults.pm).

This feature can be used in three different ways:

1. Incoming Emails:

A dedicated Postmaster filter ('PostMaster::PreFilterModule###000-SMIMEFetchFromCustomer' in SysConfig) will extract the email address of each incoming email and will try to find the email address is the list of customers, if found it will try get the S/MIME certificate from customer user attributes, if a certificate is found it will try to import it (unless it was already imported).

2. Specific email address or all customers:

   The console command 'Maint::SMIME::CustomerCertificate::Fetch' can be used to import the S/MIME certificate of one customer email address as:

   ```
   shell> perl /opt/otrs/bin/otrs.Console.pl Maint::SMIME::CustomerCertificate::Fetch --
   email customer@example.com
   ```

   In this case the console command will try to match the supplied email address with one of the customer users, if found it will try add to the system the S/MIME certificate found in customer user properties (if the certificate is not already added).

   The same console command can be used to import the S/MIME certificates of all customer users (limited to 'CustomerUserSearchListLimit' property from the customer user backend). This option is discouraged specially for systems with a large number of customer users as it might require too much time to execute and depending on the limit it might be possible that not all customer certificates will be fetch, execute the console command in this mode as:

   ```
   shell> perl /opt/otrs/bin/otrs.Console.pl Maint::SMIME::CustomerCertificate::Fetch --add-
   all
   ```

   For this option the console command will query the customer user backends to get all possible customers and for each it will check if there is a S/MIME certificate, if a certificate it found, it will try to add it to the system (if the certificate is not already added).

3. Renew existing certificates:

   Another console command 'Maint::SMIME::CustomerCertificate::renew' can be used to check for all the existing certificates in the system, this verifies that the existing certificates from customer users matches the ones that are retrieved by the customer user properties, any new certificate in the customer user backend will be added into the system (no certificates are deleted in this process).

   This console command is executed once a day by the OTRS daemon automatically with the task 'Daemon::SchedulerCronTaskManager::Task###RenewCustomerSMIMECertificates' (as seen in SysConfig), but it can be also executed manually on demand as:

   ```
   shell> perl /opt/otrs/bin/otrs.Console.pl Maint::SMIME::CustomerCertificate::Renew
   ```

5. Using External backends

5.1. Customer Data

OTRS works with many customer data attributes such as username, email address, phone number, etc. These attributes are displayed in both the Agent and the Customer frontends, and also used for the authentication of customers.

Customer data used or displayed within OTRS is highly customizable. The following information is however always needed for customer authentication:

- User login
- Email address
- Customer ID

Use the following SysConfig parameters if you want to display customer information in your agent interface.

```
    # Ticket::Frontend::CustomerInfo*
    # (show customer info on Compose (Phone and Email), Zoom and
    # Queue view)
    $Self->{'Ticket::Frontend::CustomerInfoCompose'} = 1;
    $Self->{'Ticket::Frontend::CustomerInfoZoom'} = 1;
```

Script: SysConfig configuration parameters.

5.2. Customer User Backend

You can use two types of customer backends, DB and LDAP. If you already have another customer backend (e.g. SAP), it is of course possible to write a module that uses it.

5.2.1. Database (Default)

The Example below shows the configuration of a DB customer backend, which uses customer data stored in the OTRS database.

Example 4.7. Configuring a DB customer backend

```
# CustomerUser (customer database backend and settings)
$Self->{CustomerUser} = {
    Name => 'Database Datasource',
    Module => 'Kernel::System::CustomerUser::DB',
    Params => {
        # if you want to use an external database, add the required settings
#            DSN => 'DBI:odbc:yourdsn',
#            Type => 'mssql', # only for ODBC connections
#            DSN => 'DBI:mysql:database=customerdb;host=customerdbhost',
#            User => '',
#            Password => '',
            Table => 'customer_user',

            # CaseSensitive will control if the SQL statements need LOWER()
            #   function calls to work case insensitively. Setting this to
            #   1 will improve performance dramatically on large databases.
            CaseSensitive => 0,
        },
# customer unique id
CustomerKey => 'login',

# customer #
CustomerID => 'customer_id',
CustomerValid => 'valid_id',
    CustomerUserListFields => ['first_name', 'last_name', 'email'],
    CustomerUserSearchFields => ['login', 'last_name', 'customer_id'],
    CustomerUserSearchPrefix => '',
    CustomerUserSearchSuffix => '*',
    CustomerUserSearchListLimit => 250,
    CustomerUserPostMasterSearchFields => ['email'],
    CustomerUserNameFields => ['title','first_name','last_name'],
    CustomerUserEmailUniqCheck => 1,
#    # show not own tickets in customer panel, CompanyTickets
#    CustomerUserExcludePrimaryCustomerID => 0,
```

```
#    # generate auto logins
#    AutoLoginCreation => 0,
#    AutoLoginCreationPrefix => 'auto',
#    # admin can change customer preferences
#    AdminSetPreferences => 1,
#    # cache time to live in sec. - cache any database queries
#    CacheTTL => 0,
#    # just a read only source
#    ReadOnly => 1,
    Map => [
        # note: Login, Email and CustomerID needed!
        # var, frontend, storage, shown (1=always,2=lite), required, storage-type, http-
link, readonly, http-link-target
        [ 'UserTitle',      'Title',      'title',      1, 0, 'var', '', 0 ],
        [ 'UserFirstname',  'Firstname',  'first_name', 1, 1, 'var', '', 0 ],
        [ 'UserLastname',   'Lastname',   'last_name',  1, 1, 'var', '', 0 ],
        [ 'UserLogin',      'Username',   'login',      1, 1, 'var', '', 0 ],
        [ 'UserPassword',   'Password',   'pw',         0, 0, 'var', '', 0 ],
        [ 'UserEmail',      'Email',      'email',      1, 1, 'var', '', 0 ],

#        [ 'UserEmail',      'Email', 'email',           1, 1, 'var', '[% Env("CGIHandle")
 %]?Action=AgentTicketCompose&ResponseID=1&TicketID=[% Data.TicketID %]&ArticleID=[%
 Data.ArticleID %]', 0 ],
        [ 'UserCustomerID', 'CustomerID', 'customer_id', 0, 1, 'var', '', 0 ],

#        [ 'UserCustomerIDs', 'CustomerIDs', 'customer_ids', 1, 0, 'var', '', 0 ],
        [ 'UserPhone',        'Phone',       'phone',        1, 0, 'var', '', 0 ],
        [ 'UserFax',          'Fax',         'fax',          1, 0, 'var', '', 0 ],
        [ 'UserMobile',       'Mobile',      'mobile',       1, 0, 'var', '', 0 ],
        [ 'UserStreet',       'Street',      'street',       1, 0, 'var', '', 0 ],
        [ 'UserZip',          'Zip',         'zip',          1, 0, 'var', '', 0 ],
        [ 'UserCity',         'City',        'city',         1, 0, 'var', '', 0 ],
        [ 'UserCountry',      'Country',     'country',      1, 0, 'var', '', 0 ],
        [ 'UserComment',      'Comment',     'comments',     1, 0, 'var', '', 0 ],
        [ 'ValidID',          'Valid',       'valid_id',     0, 1, 'int', '', 0 ],
    ],
    # default selections
    Selections => {
        UserTitle => {
            'Mr.' => 'Mr.',
            'Mrs.' => 'Mrs.',
        },
    },
};
```

If you want to customize the customer data, change the column headers or add new ones to the customer_user table in the OTRS database. As an example, the script below shows how to add a new field for room number.

```
linux:~# mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 116 to server version: 5.0.18-Debian_7-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use otrs;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> ALTER TABLE customer_user ADD room VARCHAR (250);
Query OK, 1 rows affected (0.01 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> quit
Bye
linux:~#
```

Script: Adding a room field to the customer_user table.

Now add the new column to the MAP array in `Kernel/Config.pm`, as shown in the following script.

```
   # var, frontend, storage, shown (1=always,2=lite), required, storage-type, http-link,
readonly
   [...]
   [ 'UserRoom',      'Room',      'room',       0, 1, 'var', '', 0 ],
```

Script: Adding a room field to the Kernel/Config.pm file.

It is also possible to edit all of this customer information via the Customers link in the Agent interface.

5.2.1.1. Customer with Multiple IDs (Company Tickets)

It is possible to assign more than one customer ID to a customer. This can be useful if a customer must access tickets of other customers, e.g. a supervisor wants to watch the tickets of his assistants. If a customer can access the tickets of another customer, the company ticket feature of OTRS is used. Company tickets can be accessed via the "Company Tickets" link in the customer panel.

To use company tickets, a new column with the IDs that should be accessible for a customer, has to be added to the customer_user table in the OTRS database (see Script below).

```
linux:~# mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 124 to server version: 5.0.18-Debian_7-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use otrs;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> ALTER TABLE customer_user ADD customer_ids VARCHAR (250);
Query OK, 1 rows affected (0.02 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> quit
Bye
linux:~#
```

Script: Adding a customer_ids field to the customer_user table.

Now the new column has to be added to the MAP array in `Kernel/Config.pm`, as shown in the script below.

```
   # var, frontend, storage, shown (1=always,2=lite), required, storage-type, http-link,
readonly
   [...]
   [ 'UserCustomerIDs', 'CustomerIDs', 'customer_ids', 1, 0, 'var', '', 0 ],
```

Script: Adding a UserCustomerIDs field to the Kernel/Config.pm file.

Now, the new column for the multiple customer IDs can be edited via the Agent interface, in the section for the customer management.

To ensure that one customer can access the tickets of other customers, add the IDs of these other users into the new field for the multiple customer IDs. Each ID has to be separated by a semicolon (see Example below).

Example 4.8. Using Company Tickets with a DB Backend

The customers A, B and C exist in your system, and A wants to have access to the tickets of B and C via the customer panel. B and C should have no access to the tickets of other users.

To realize this setup, change the customer_user table and the mapping in `Kernel/Config.pm` as described above. Then load the settings for customer A via the Customers link in the Agent interface or via the Admin page. If the settings are displayed, add into the field for CustomerIDs the values "B;C;".

5.2.2. LDAP

If you have an LDAP directory with your customer data, you can use it as the customer backend with OTRS, as shown in Example below.

Example 4.9. Configuring an LDAP customer backend

```
# CustomerUser
# (customer ldap backend and settings)
$Self->{CustomerUser} = {
    Name => 'LDAP Data Source',
    Module => 'Kernel::System::CustomerUser::LDAP',
    Params => {
        # ldap host
        Host => 'bay.csuhayward.edu',
        # ldap base dn
        BaseDN => 'ou=seas,o=csuh',
        # search scope (one|sub)
        SSCOPE => 'sub',
        # The following is valid but would only be necessary if the
        # anonymous user does NOT have permission to read from the LDAP tree
        UserDN => '',
        UserPw => '',
        # in case you want to add always one filter to each ldap query, use
        # this option. e. g. AlwaysFilter => '(mail=*)' or AlwaysFilter =>
 '(objectclass=user)'
        AlwaysFilter => '',
            # if the charset of your ldap server is iso-8859-1, use this:
#            SourceCharset => 'iso-8859-1',

            # Net::LDAP new params (if needed - for more info see perldoc Net::LDAP)
            Params => {
                port => 389,
                timeout => 120,
                async => 0,
                version => 3,
            },
    },
    # customer unique id
    CustomerKey => 'uid',
    # customer #
    CustomerID => 'mail',
    CustomerUserListFields => ['cn', 'mail'],
    CustomerUserSearchFields => ['uid', 'cn', 'mail'],
    CustomerUserSearchPrefix => '',
    CustomerUserSearchSuffix => '*',
    CustomerUserSearchListLimit => 250,
    CustomerUserPostMasterSearchFields => ['mail'],
    CustomerUserNameFields => ['givenname', 'sn'],
    # show not own tickets in customer panel, CompanyTickets
    CustomerUserExcludePrimaryCustomerID => 0,
    # add an ldap filter for valid users (expert setting)
#    CustomerUserValidFilter => '(!(description=locked))',
    # administrator can't change customer preferences
    AdminSetPreferences => 0,
#    # cache time to live in sec. - cache any database queries
```

```
#    CacheTTL => 0,
    Map => [
        # note: Login, Email and CustomerID are mandatory!
        # var, frontend, storage, shown (1=always,2=lite), required, storage-type, http-
link, readonly
        [ 'UserTitle',      'Title',      'title',           1, 0, 'var', '', 0 ],
        [ 'UserFirstname',  'Firstname',  'givenname',       1, 1, 'var', '', 0 ],
        [ 'UserLastname',   'Lastname',   'sn',              1, 1, 'var', '', 0 ],
        [ 'UserLogin',      'Username',   'uid',             1, 1, 'var', '', 0 ],
        [ 'UserEmail',      'Email',      'mail',            1, 1, 'var', '', 0 ],
        [ 'UserCustomerID', 'CustomerID', 'mail',            0, 1, 'var', '', 0 ],
#        [ 'UserCustomerIDs', 'CustomerIDs', 'second_customer_ids', 1, 0, 'var', '', 0 ],
        [ 'UserPhone',      'Phone',      'telephonenumber', 1, 0, 'var', '', 0 ],
        [ 'UserAddress',    'Address',    'postaladdress',   1, 0, 'var', '', 0 ],
        [ 'UserComment',    'Comment',    'description',     1, 0, 'var', '', 0 ],
    ],
};
```

If additional customer attributes are stored in your LDAP directory, such as a manager's name, a mobile phone number, or a department, and if you want to display this information in OTRS, just expand the MAP array in `Kernel/Config.pm` with the entries for these attributes, as shown in the following script.

```
   # var, frontend, storage, shown (1=always,2=lite), required, storage-type, http-link,
readonly
   [...]
   [ 'UserPhone',      'Phone',      'telephonenumber', 1, 0, 'var', '', 0 ],
```

Script: Adding new fields to the Kernel/Config.pm file.

5.2.2.1. Customer with Multiple IDs (Company Tickets)

It is possible to assign more than one Customer ID to a customer, when using an LDAP backend. To use company tickets, a new field has to be added to the LDAP directory that contains the IDs accessible by the customer.

If the new field in the LDAP directory has been created, the new entry has to be added to the MAP array in `Kernel/Config.pm`, as shown in the script below.

```
   # var, frontend, storage, shown (1=always,2=lite), required, storage-type, http-link,
readonly
   [...]
   [ 'UserCustomerIDs', 'CustomerIDs', 'customer_ids', 1, 0, 'var', '', 0 ],
```

Script: Maping new fields to the Kernel/Config.pm file.

The field for the multiple customer IDs has to be edited directly in the LDAP directory. OTRS can only read from LDAP, not write to it.

To ensure access by a customer to the tickets of other customers, add the customer IDs of the customers whose tickets should be accessed to the new field in your LDAP directory. Each ID has to be separated by a semicolon (see Example below).

Example 4.10. Using Company tickets with an LDAP backend

The customers A, B and C exist in your system, and A wants to have access to the tickets of B and C via the customer panel. B and C should have no access to the tickets of other users.

To realize this setup, change the LDAP directory and the mapping in `Kernel/Config.pm` as described above. Then add into the field for CustomerIDs the values "B;C;" for customer A in your LDAP directory.

5.2.3. Using More than One Customer Backend with OTRS

If you want to utilize more than one customer data source used with OTRS (e.g. an LDAP and a database backend), the CustomerUser config parameter should be expanded with a number, e.g. "CustomerUser1", "CustomerUser2" (see Example below).

Example 4.11. Using more than one customer backend with OTRS

The following configuration example shows usage of both an LDAP and a database customer backend with OTRS.

```
# 1. Customer user backend: DB
# (customer database backend and settings)
$Self->{CustomerUser1} = {
    Name => 'Customer Database',
    Module => 'Kernel::System::CustomerUser::DB',
    Params => {
        # if you want to use an external database, add the
        # required settings
#        DSN => 'DBI:odbc:yourdsn',
#        Type => 'mssql', # only for ODBC connections
#        DSN => 'DBI:mysql:database=customerdb;host=customerdbhost',
#        User => '',
#        Password => '',
        Table => 'customer_user',
    },
    # customer unique id
    CustomerKey => 'login',
    # customer #
    CustomerID => 'customer_id',
    CustomerValid => 'valid_id',
    CustomerUserListFields => ['first_name', 'last_name', 'email'],
    CustomerUserSearchFields => ['login', 'last_name', 'customer_id'],
    CustomerUserSearchPrefix => '',
    CustomerUserSearchSuffix => '*',
    CustomerUserSearchListLimit => 250,
    CustomerUserPostMasterSearchFields => ['email'],
    CustomerUserNameFields => ['title','first_name','last_name'],
    CustomerUserEmailUniqCheck => 1,
#    # show not own tickets in customer panel, CompanyTickets
#    CustomerUserExcludePrimaryCustomerID => 0,
#    # generate auto logins
#    AutoLoginCreation => 0,
#    AutoLoginCreationPrefix => 'auto',
#    # admin can change customer preferences
#    AdminSetPreferences => 1,
#    # cache time to live in sec. - cache any database queries
#    CacheTTL => 0,
#    # just a read only source
#    ReadOnly => 1,
    Map => [

        # note: Login, Email and CustomerID needed!
        # var, frontend, storage, shown (1=always,2=lite), required, storage-type, http-
link, readonly, http-link-target
        [ 'UserTitle',      'Title',      'title',        1, 0, 'var', '', 0 ],
        [ 'UserFirstname',  'Firstname',  'first_name',   1, 1, 'var', '', 0 ],
        [ 'UserLastname',   'Lastname',   'last_name',    1, 1, 'var', '', 0 ],
        [ 'UserLogin',      'Username',   'login',        1, 1, 'var', '', 0 ],
        [ 'UserPassword',   'Password',   'pw',           0, 0, 'var', '', 0 ],
        [ 'UserEmail',      'Email',      'email',        1, 1, 'var', '', 0 ],
        [ 'UserCustomerID', 'CustomerID', 'customer_id',  0, 1, 'var', '', 0 ],
        [ 'UserPhone',      'Phone',      'phone',        1, 0, 'var', '', 0 ],
        [ 'UserFax',        'Fax',        'fax',          1, 0, 'var', '', 0 ],
        [ 'UserMobile',     'Mobile',     'mobile',       1, 0, 'var', '', 0 ],
        [ 'UserStreet',     'Street',     'street',       1, 0, 'var', '', 0 ],
        [ 'UserZip',        'Zip',        'zip',          1, 0, 'var', '', 0 ],
```

```
        [ 'UserCity',       'City',       'city',         1, 0, 'var', '', 0 ],
        [ 'UserCountry',    'Country',    'country',      1, 0, 'var', '', 0 ],
        [ 'UserComment',    'Comment',    'comments',     1, 0, 'var', '', 0 ],
        [ 'ValidID',        'Valid',      'valid_id',     0, 1, 'int', '', 0 ],
    ],
    # default selections
    Selections => {
        UserTitle => {
            'Mr.' => 'Mr.',
            'Mrs.' => 'Mrs.',
        },
    },
};

# 2. Customer user backend: LDAP
# (customer ldap backend and settings)
$Self->{CustomerUser2} = {
    Name => 'LDAP Datasource',
    Module => 'Kernel::System::CustomerUser::LDAP',
    Params => {
        # ldap host
        Host => 'bay.csuhayward.edu',
        # ldap base dn
        BaseDN => 'ou=seas,o=csuh',
        # search scope (one|sub)
        SSCOPE => 'sub',
        # The following is valid but would only be necessary if the
        # anonymous user does NOT have permission to read from the LDAP tree
        UserDN => '',
        UserPw => '',
        # in case you want to add always one filter to each ldap query, use
        # this option. e. g. AlwaysFilter => '(mail=*)' or AlwaysFilter =>
 '(objectclass=user)'
        AlwaysFilter => '',
        # if the charset of your ldap server is iso-8859-1, use this:
#        SourceCharset => 'iso-8859-1',

        # Net::LDAP new params (if needed - for more info see perldoc Net::LDAP)
        Params => {
            port => 389,
            timeout => 120,
            async => 0,
            version => 3,
        },
    },
    # customer unique id
    CustomerKey => 'uid',
    # customer #
    CustomerID => 'mail',
    CustomerUserListFields => ['cn', 'mail'],
    CustomerUserSearchFields => ['uid', 'cn', 'mail'],
    CustomerUserSearchPrefix => '',
    CustomerUserSearchSuffix => '*',
    CustomerUserSearchListLimit => 250,
    CustomerUserPostMasterSearchFields => ['mail'],
    CustomerUserNameFields => ['givenname', 'sn'],
    # show not own tickets in customer panel, CompanyTickets
    CustomerUserExcludePrimaryCustomerID => 0,
    # add a ldap filter for valid users (expert setting)
#    CustomerUserValidFilter => '(!(description=locked))',
    # admin can't change customer preferences
    AdminSetPreferences => 0,
    Map => [
        # note: Login, Email and CustomerID needed!
        # var, frontend, storage, shown (1=always,2=lite), required, storage-type, http-
link, readonly
        [ 'UserTitle',      'Title',      'title',          1, 0, 'var', '', 0 ],
        [ 'UserFirstname',  'Firstname',  'givenname',      1, 1, 'var', '', 0 ],
        [ 'UserLastname',   'Lastname',   'sn',             1, 1, 'var', '', 0 ],
        [ 'UserLogin',      'Username',   'uid',            1, 1, 'var', '', 0 ],
        [ 'UserEmail',      'Email',      'mail',           1, 1, 'var', '', 0 ],
        [ 'UserCustomerID', 'CustomerID', 'mail',           0, 1, 'var', '', 0 ],
```

```
#        [ 'UserCustomerIDs', 'CustomerIDs', 'second_customer_ids', 1, 0, 'var', '', 0 ],
        [ 'UserPhone',      'Phone',       'telephonenumber', 1, 0, 'var', '', 0 ],
        [ 'UserAddress',    'Address',     'postaladdress',   1, 0, 'var', '', 0 ],
        [ 'UserComment',    'Comment',     'description',     1, 0, 'var', '', 0 ],
    ],
};
```

It is possible to integrate up to 10 different customer backends. Use the customer management interface in OTRS to view or edit (assuming write access is enabled) all customer data.

5.2.4. Storing CustomerUser Data in Dynamic Fields

Sometimes it can be useful to also store CustomerUser data directly in dynamic fields of a ticket, for example to create special statistics on this data.

The dynamic field values are set when a ticket is created or when the customer of a ticket is changed. The values of the dynamic fields are taken from the customer data. This works for all backends, but is especially useful for LDAP-backends.

To activate this optional feature of OTRS, please activate the settings "Ticket::EventModulePost###930-DynamicFieldFromCustomerUser" and "DynamicFieldFromCustomerUser::Mapping". The latter setting contains the configuration of which CustomerUser field entry should be stored in which ticket dynamic field. The fields must be present in the system and should be enabled for AgentTicketFreeText, so that they can be set manually. They mustn't be enabled for AgentTicketPhone, AgentTicketEmail and AgentTicketCustomer. If they were, they would have precedence over the automatically set values.

5.3. Backends to Authenticate Agents and Customers

OTRS offers the option to authenticate agents and customers against different backends.

5.3.1. Authentication backends for Agents

5.3.1.1. DB (Default)

The backend to authenticate agents which is used by default is the OTRS database. Agents can be added and edited via the agent management interface in the Admin page (see Example below).

Example 4.12. Authenticate agents against a DB backend

```
    $Self->{'AuthModule'} = 'Kernel::System::Auth::DB';
```

5.3.1.2. LDAP

If an LDAP directory has all your agent data stored, you can use the LDAP module to authenticate your users in OTRS (see Example below). This module has only read access to the LDAP tree, which means that you cannot edit your user data via the agent management interface.

Example 4.13. Authenticate agents against an LDAP backend

```
# This is an example configuration for an LDAP auth. backend.
# (Make sure Net::LDAP is installed!)
$Self->{'AuthModule'} = 'Kernel::System::Auth::LDAP';
$Self->{'AuthModule::LDAP::Host'} = 'ldap.example.com';
```

```
$Self->{'AuthModule::LDAP::BaseDN'} = 'dc=example,dc=com';
$Self->{'AuthModule::LDAP::UID'} = 'uid';

# Check if the user is allowed to auth in a posixGroup
# (e. g. user needs to be in a group xyz to use otrs)
$Self->{'AuthModule::LDAP::GroupDN'} = 'cn=otrsallow,ou=posixGroups,dc=example,dc=com';
$Self->{'AuthModule::LDAP::AccessAttr'} = 'memberUid';
# for ldap posixGroups objectclass (just uid)
#  $Self->{'AuthModule::LDAP::UserAttr'} = 'UID';
# for non ldap posixGroups objectclass (with full user dn)
#  $Self->{'AuthModule::LDAP::UserAttr'} = 'DN';

# The following is valid but would only be necessary if the
# anonymous user do NOT have permission to read from the LDAP tree
$Self->{'AuthModule::LDAP::SearchUserDN'} = '';
$Self->{'AuthModule::LDAP::SearchUserPw'} = '';

# in case you want to add always one filter to each ldap query, use
# this option. e. g. AlwaysFilter => '(mail=*)' or AlwaysFilter => '(objectclass=user)'
$Self->{'AuthModule::LDAP::AlwaysFilter'} = '';

# in case you want to add a suffix to each login name, then
# you can use this option. e. g. user just want to use user but
# in your ldap directory exists user@domain.com
#    $Self->{'AuthModule::LDAP::UserSuffix'} = '@domain.com';

# Net::LDAP new params (if needed - for more info see perldoc Net::LDAP)
$Self->{'AuthModule::LDAP::Params'} = {
    port => 389,
    timeout => 120,
    async => 0,
    version => 3,
};
```

The configuration parameters shown in the script below can be used to synchronize the user data from your LDAP directory into your local OTRS database. This reduces the number of requests to your LDAP server and speeds up the authentication with OTRS. The data synchronization is done when the agent authenticates the first time. Although the data can be syncronized into the local OTRS database, the LDAP directory is the last instance for the authentication, so an inactive user in the LDAP tree can't authenticate to OTRS, even when the account data is already stored in the OTRS database. The agent data in the LDAP directory can't be edited via the web interface of OTRS, so the data has to be managed directly in the LDAP tree.

```
# defines AuthSyncBackend (AuthSyncModule) for AuthModule
# if this key exists and is empty, there won't be a sync.
# example values: AuthSyncBackend, AuthSyncBackend2
$Self->{'AuthModule::UseSyncBackend'} = 'AuthSyncBackend';

# agent data sync against ldap
$Self->{'AuthSyncModule'} = 'Kernel::System::Auth::Sync::LDAP';
$Self->{'AuthSyncModule::LDAP::Host'} = 'ldap://ldap.example.com/';
$Self->{'AuthSyncModule::LDAP::BaseDN'} = 'dc=otrs, dc=org';
$Self->{'AuthSyncModule::LDAP::UID'} = 'uid';
$Self->{'AuthSyncModule::LDAP::SearchUserDN'} = 'uid=sys, ou=user, dc=otrs, dc=org';
$Self->{'AuthSyncModule::LDAP::SearchUserPw'} = 'some_pass';
$Self->{'AuthSyncModule::LDAP::UserSyncMap'} = {
    # DB -> LDAP
    UserFirstname => 'givenName',
    UserLastname  => 'sn',
    UserEmail     => 'mail',
};
[...]

# AuthSyncModule::LDAP::UserSyncInitialGroups
# (sync following group with rw permission after initial create of first agent
# login)
$Self->{'AuthSyncModule::LDAP::UserSyncInitialGroups'} = [
    'users',
```

```
];
```

Script: Synchronizing the user data from the LDAP directory into the OTRS database.

Alternatively, you can use LDAP groups to determine group memberships or roles in OTRS. For more information and examples, see `Kernel/Config/Defaults.pm`. Here is an example for synchronizing from LDAP into OTRS groups.

```
# Attributes needed for group syncs
# (attribute name for group value key)
$Self->{'AuthSyncModule::LDAP::AccessAttr'} = 'memberUid';
# (select the attribute for type of group content UID/DN for full ldap name)
# $Self->{'AuthSyncModule::LDAP::UserAttr'} = 'UID';
# $Self->{'AuthSyncModule::LDAP::UserAttr'} = 'DN';

AuthSyncModule::LDAP::UserSyncGroupsDefinition
# (If "LDAP" was selected for AuthModule and you want to sync LDAP
# groups to otrs groups, define the following.)
$Self->{'AuthSyncModule::LDAP::UserSyncGroupsDefinition'} = {
    # your ldap group
    'cn=agent,o=otrs' => {
        # otrs group(s)
        'admin' => {
            # permission
            rw => 1,
            ro => 1,
        },
        'faq' => {
            rw => 0,
            ro => 1,
        },
    },
    'cn=agent2,o=otrs' => {
        'users' => {
            rw => 1,
            ro => 1,
        },
    }
};
```

5.3.1.3. HTTPBasicAuth for Agents

If you want to implement a "single sign on" solution for all your agents, you can use HTTP basic authentication (for all your systems) and the HTTPBasicAuth module for OTRS (see Example below).

Example 4.14. Authenticate Agents using HTTPBasic

```
# This is an example configuration for an apache ($ENV{REMOTE_USER})
# auth. backend. Use it if you want to have a singe login through
# apache http-basic-auth
$Self->{'AuthModule'} = 'Kernel::System::Auth::HTTPBasicAuth';

# Note:
#
# If you use this module, you should use as fallback
# the following configuration settings if the user is not authorized
# apache ($ENV{REMOTE_USER})
$Self->{LoginURL} = 'http://host.example.com/not-authorised-for-otrs.html';
$Self->{LogoutURL} = 'http://host.example.com/thanks-for-using-otrs.html';
```

5.3.1.4. Radius

The configuration parameters shown in Example below can be used to authenticate agents against a Radius server.

Example 4.15. Authenticate Agents against a Radius backend

```
# This is example configuration to auth. agents against a radius server
$Self->{'AuthModule'} = 'Kernel::System::Auth::Radius';
$Self->{'AuthModule::Radius::Host'} = 'radiushost';
$Self->{'AuthModule::Radius::Password'} = 'radiussecret';
```

5.3.2. Authentication Backends for Customers

5.3.2.1. Database (Default)

The default user authentication backend for customers in OTRS is the OTRS database. With this backend, all customer data can be edited via the web interface of OTRS (see Example below).

Example 4.16. Customer user authentication against a DB backend

```
# This is the auth. module against the otrs db
$Self->{'Customer::AuthModule'} = 'Kernel::System::CustomerAuth::DB';
$Self->{'Customer::AuthModule::DB::Table'} = 'customer_user';
$Self->{'Customer::AuthModule::DB::CustomerKey'} = 'login';
$Self->{'Customer::AuthModule::DB::CustomerPassword'} = 'pw';
#$Self->{'Customer::AuthModule::DB::DSN'} =
 "DBI:mysql:database=customerdb;host=customerdbhost";
#$Self->{'Customer::AuthModule::DB::User'} = "some_user";
#$Self->{'Customer::AuthModule::DB::Password'} = "some_password";
```

5.3.2.2. LDAP

If you have an LDAP directory with all your customer data, you can use the LDAP module to authenticate your customers to OTRS (see Example below). Because this module has only read-access to the LDAP backend, it is not possible to edit the customer data via the OTRS web interface.

Example 4.17. Customer user authentication against an LDAP backend

```
# This is an example configuration for an LDAP auth. backend.
# (make sure Net::LDAP is installed!)
$Self->{'Customer::AuthModule'} = 'Kernel::System::CustomerAuth::LDAP';
$Self->{'Customer::AuthModule::LDAP::Host'} = 'ldap.example.com';
$Self->{'Customer::AuthModule::LDAP::BaseDN'} = 'dc=example,dc=com';
$Self->{'Customer::AuthModule::LDAP::UID'} = 'uid';

# Check if the user is allowed to auth in a posixGroup
# (e. g. user needs to be in a group xyz to use otrs)
$Self->{'Customer::AuthModule::LDAP::GroupDN'} =
 'cn=otrsallow,ou=posixGroups,dc=example,dc=com';
$Self->{'Customer::AuthModule::LDAP::AccessAttr'} = 'memberUid';
# for ldap posixGroups objectclass (just uid)
$Self->{'Customer::AuthModule::LDAP::UserAttr'} = 'UID';
# for non ldap posixGroups objectclass (full user dn)
#$Self->{'Customer::AuthModule::LDAP::UserAttr'} = 'DN';

# The following is valid but would only be necessary if the
# anonymous user does NOT have permission to read from the LDAP tree
$Self->{'Customer::AuthModule::LDAP::SearchUserDN'} = '';
$Self->{'Customer::AuthModule::LDAP::SearchUserPw'} = '';

# in case you want to add always one filter to each ldap query, use
# this option. e. g. AlwaysFilter => '(mail=*)' or AlwaysFilter => '(objectclass=user)'
$Self->{'Customer::AuthModule::LDAP::AlwaysFilter'} = '';

# in case you want to add a suffix to each customer login name, then
```

```
# you can use this option. e. g. user just want to use user but
# in your ldap directory exists user@domain.com
#$Self->{'Customer::AuthModule::LDAP::UserSuffix'} = '@domain.com';

# Net::LDAP new params (if needed - for more info see perldoc Net::LDAP)
$Self->{'Customer::AuthModule::LDAP::Params'} = {
    port => 389,
    timeout => 120,
    async => 0,
    version => 3,
};
```

5.3.2.3. HTTPBasicAuth for Customers

If you want to implement a "single sign on" solution for all your customer users, you can use HTTPBasic authentication (for all your systems) and use the HTTPBasicAuth module with OTRS (no login is needed with OTRS any more). See Example below.

Example 4.18. Customer user authentication with HTTPBasic

```
# This is an example configuration for an apache ($ENV{REMOTE_USER})
# auth. backend. Use it if you want to have a singe login through
# apache http-basic-auth
$Self->{'Customer::AuthModule'} = 'Kernel::System::CustomerAuth::HTTPBasicAuth';

# Note:
# If you use this module, you should use the following
# config settings as fallback, if user isn't login through
# apache ($ENV{REMOTE_USER})
$Self->{CustomerPanelLoginURL} = 'http://host.example.com/not-authorised-for-otrs.html';
$Self->{CustomerPanelLogoutURL} = 'http://host.example.com/thanks-for-using-otrs.html';
```

5.3.2.4. Radius

The settings shown in Example below can be used to authenticate your customers against a Radius server.

Example 4.19. Customer user authentication against a Radius backend

```
# This is a example configuration to auth. customer against a radius server
$Self->{'Customer::AuthModule'} = 'Kernel::System::Auth::Radius';
$Self->{'Customer::AuthModule::Radius::Host'} = 'radiushost';
$Self->{'Customer::AuthModule::Radius::Password'} = 'radiussecret';
```

5.4. Customizing the Customer Self-Registration

It is possible to customize the self-registration for new customers, accessible via the customer.pl panel. New optional or required fields, like room number, address or state can be added.

The following example shows how you can specify a required field in the customer database, in this case to store the room number of a customer.

5.4.1. Customizing the Web Interface

To display the new field for the room number in the customer.pl web interface, the .dtl file responsible for the layout in this interface has to be modified. Edit the `Kernel/Output/HTML/Standard/CustomerLogin.dtl` file, adding the new field around line 80 (see Script below).

```
[...]
<div class="NewLine">
    <label for="Room">[% Translate("Room{CustomerUser}") | html %]</label>
    <input title="[% Translate("Room Number") | html %]" name="Room" type="text"
 id="UserRoom" maxlength="50" />
</div>
[...]
```

Script: Displaying a new field in the web interface.

5.4.2. Customer Mapping

In the next step, the customer mapping has to be expanded with the new entry for the room number. To ensure that the changes are not lost after an update, put the "CustomerUser" settings from the `Kernel/Config/Defaults.pm` into the `Kernel/Config.pm`. Now change the MAP array and add the new room number field, as shown in the script below.

```
# CustomerUser
# (customer database backend and settings)
$Self->{CustomerUser} = {
    Name => 'Database Backend',
    Module => 'Kernel::System::CustomerUser::DB',
    Params => {
        # if you want to use an external database, add the
        # required settings
#        DSN => 'DBI:odbc:yourdsn',
#        Type => 'mssql', # only for ODBC connections
#        DSN => 'DBI:mysql:database=customerdb;host=customerdbhost',
#        User => '',
#        Password => '',
        Table => 'customer_user',
    },
    # customer unique id
    CustomerKey => 'login',
    # customer #
    CustomerID => 'customer_id',
    CustomerValid => 'valid_id',
    CustomerUserListFields => ['first_name', 'last_name', 'email'],
#    CustomerUserListFields => ['login', 'first_name', 'last_name', 'customer_id', 'email'],
    CustomerUserSearchFields => ['login', 'last_name', 'customer_id'],
    CustomerUserSearchPrefix => '',
    CustomerUserSearchSuffix => '*',
    CustomerUserSearchListLimit => 250,
    CustomerUserPostMasterSearchFields => ['email'],
    CustomerUserNameFields => ['title', 'first_name', 'last_name'],
    CustomerUserEmailUniqCheck => 1,
#    # show not own tickets in customer panel, CompanyTickets
#    CustomerUserExcludePrimaryCustomerID => 0,
#    # generate auto logins
#    AutoLoginCreation => 0,
#    AutoLoginCreationPrefix => 'auto',
#    # admin can change customer preferences
#    AdminSetPreferences => 1,
#    # cache time to live in sec. - cache database queries
#    CacheTTL => 0,
#    # just a read only source
#    ReadOnly => 1,
    Map => [

        # note: Login, Email and CustomerID needed!
        # var, frontend, storage, shown (1=always,2=lite), required, storage-type, http-
link, readonly, http-link-target
        [ 'UserTitle',      'Title',      'title',       1, 0, 'var', '', 0 ],
        [ 'UserFirstname',  'Firstname',  'first_name',  1, 1, 'var', '', 0 ],
        [ 'UserLastname',   'Lastname',   'last_name',   1, 1, 'var', '', 0 ],
        [ 'UserLogin',      'Username',   'login',       1, 1, 'var', '', 0 ],
        [ 'UserPassword',   'Password',   'pw',          0, 0, 'var', '', 0 ],
```

```
        [ 'UserEmail',      'Email',      'email',       1, 1, 'var', '', 0 ],
        [ 'UserCustomerID', 'CustomerID', 'customer_id', 0, 1, 'var', '', 0 ],
        [ 'UserPhone',      'Phone',      'phone',       1, 0, 'var', '', 0 ],
        [ 'UserFax',        'Fax',        'fax',         1, 0, 'var', '', 0 ],
        [ 'UserMobile',     'Mobile',     'mobile',      1, 0, 'var', '', 0 ],
        [ 'UserRoom',       'Room',       'room',        1, 0, 'var', '', 0 ],
        [ 'UserStreet',     'Street',     'street',      1, 0, 'var', '', 0 ],
        [ 'UserZip',        'Zip',        'zip',         1, 0, 'var', '', 0 ],
        [ 'UserCity',       'City',       'city',        1, 0, 'var', '', 0 ],
        [ 'UserCountry',    'Country',    'country',     1, 0, 'var', '', 0 ],
        [ 'UserComment',    'Comment',    'comments',    1, 0, 'var', '', 0 ],
        [ 'ValidID',        'Valid',      'valid_id',    0, 1, 'int', '', 0 ],
    ],
    # default selections
    Selections => {
        UserTitle => {
            'Mr.' => 'Mr.',
            'Mrs.' => 'Mrs.',
        },
    },
};
```

Script: Changing the map array.

5.4.3. Customizing the `customer_user` Table in the OTRS DB

The last step is to add the new room number column to the customer_user table in the OTRS database (see Script below). In this column, the entries for the room numbers will be stored.

```
linux:~# mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6 to server version: 5.0.18-Debian_7-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use otrs;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> ALTER TABLE customer_user ADD room VARCHAR (200);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> quit
Bye
linux:~#
```

Script: Adding a new column to the customer_user table.

Now the new field for the room should be displayed in the Customer Information panel if filled, and in the Customer User administration screens. Also, new customers should have to insert their room number if they register a new account.

6. Ticket Settings

6.1. Ticket States

6.1.1. Predefined states

OTRS allows you to change predefined ticket states and their types, or even add new ones. Two attributes are important for a state: the state name and the state type.

The default states of OTRS are: 'closed successful', 'closed unsuccessful', 'merged', 'new', 'open', 'pending auto close+', 'pending auto close-', 'pending reminder' and 'removed'.

6.1.1.1. New

Tickets are usually in this state when created from incoming e-mails.

6.1.1.2. Open

This is the default state for tickets assigned to queues and agents.

6.1.1.3. Pending reminder

After the pending time has expired, the ticket owner will receive a reminder email concerning the ticket. If the ticket is not locked, the reminder will be sent to all agents in the queue. Reminder tickets will only be sent out during business hours, and are repeatedly sent every 24 hours until the ticket state is changed by the agent. Time spent by the ticket in this status will still add towards the escalation time calculation.

6.1.1.4. Pending auto close-

Tickets in this status will be set to "Closed Unsuccessful" if the pending time has expired. Time spent by the ticket in this status will still add towards the escalation time calculation.

6.1.1.5. Pending auto close+

Tickets in this status will be set to "Closed Successful" if the pending time has expired. Time spent by the ticket in this status will still add towards the escalation time calculation.

6.1.1.6. Merged

This is the state for tickets that have been merged with other tickets.

6.1.1.7. Closed Successful

This is the end state for tickets that have been successfully resolved. Depending on your configuration, you may or may not be able to reopen closed tickets.

6.1.1.8. Closed Unsuccessful

This is the end state for tickets that have NOT been successfully resolved. Depending on your configuration, you may or may not be able to reopen closed tickets.

6.1.2. Customizing states

Every state has a name (state-name) and a type (state-type). Click on the States link on the Admin page and press the button "Add state" to create a new state. You can freely choose the name of a new state. The state types can not be changed via the web interface. The database has to be directly modified if you want to add new types or change existing names. The default state types should typically not be modified as this can yield unpredictable results. For instance, escalation calculations and the unlock feature are based on specific state types.

The name of an already existing state can be changed, or new states added through this screen. If the state "new" has been changed via the web interface, this change also has to be configured via the config file `Kernel/Config.pm` or via the SysConfig interface. The settings specified in the script below have to be modified to ensure that OTRS works with the changed state for "new".

```
    [...]
    # PostmasterDefaultState
    # (The default state of new tickets.) [default: new]
    $Self->{PostmasterDefaultState} = 'new';

    # CustomerDefaultState
    # (default state of new customer tickets)
    $Self->{CustomerDefaultState} = 'new';
    [...]
```

Script: Modifying the Kernel/Config.pm settings.

If a new state type should be added, the ticket_state_type table in the OTRS database needs to be modified with a database client program, as shown in the script below.

```
linux:~# mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 23 to server version: 5.0.16-Debian_1-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use otrs;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> insert into ticket_state_type (name,comments) values ('own','Own
state type');
Query OK, 1 row affected (0.00 sec)

mysql> quit
Bye
linux:~#
```

Script: Modifying the OTRS database.

Now it is possible to use the new state type you just created. After a state has been linked with this new state type, the OTRS configuration also has to be changed to ensure that the new state is usable. Just modify the following options via SysConfig:

Ticket -> Frontend::Agent::Ticket::ViewPhoneNew > AgentTicketPhone###StateDefault - to define the default next state for new phone tickets.

Ticket -> Frontend::Agent::Ticket::ViewPhoneNew > AgentTicketPhone###StateType - to define the available next states for new phone tickets.

Ticket -> Frontend::Agent::Ticket::ViewEmailNew > AgentTicketEmail###StateDefault - to define the default next state for new email tickets.

Ticket -> Frontend::Agent::Ticket::ViewEmailNew > AgentTicketEmail###StateType - to define the available next states for new email tickets.

Ticket -> Frontend::Agent::Ticket::ViewPhoneOutbound > AgentTicketPhoneOutbound###State - to define the default next state for new phone articles.

Ticket -> Frontend::Agent::Ticket::ViewPhoneOutbound > AgentTicketPhoneOutbound###StateType - to define the available next states for new phone articles.

Ticket -> Frontend::Agent::Ticket::ViewMove > AgentTicketMove###State - to define the default next state for moving a ticket.

Ticket -> Frontend::Agent::Ticket::ViewMove > AgentTicketMove###StateType - to define the available next states for moving a ticket.

Ticket -> Frontend::Agent::Ticket::ViewBounce > StateDefault - to define the default next state after bouncing a ticket.

Ticket -> Frontend::Agent::Ticket::ViewBounce > StateType - to define the available next states in the bounce screen.

Ticket -> Frontend::Agent::Ticket::ViewBulk > StateDefault - to define the default next state in a bulk action.

Ticket -> Frontend::Agent::Ticket::ViewBulk > StateType - to define the available next states in the bulk action screen.

Ticket -> Frontend::Agent::Ticket::ViewClose > StateDefault - to define the default next state after closing a ticket.

Ticket -> Frontend::Agent::Ticket::ViewClose > StateType - to define the available next states in the close screen.

Ticket -> Frontend::Agent::Ticket::ViewCompose > StateDefault - to define the default next state in the Compose (reply) screen.

Ticket -> Frontend::Agent::Ticket::ViewCompose > StateType - to define the available next states in the Compose (reply) screen.

Ticket -> Frontend::Agent::Ticket::ViewForward > StateDefault - to define the default next state after forwarding a ticket.

Ticket -> Frontend::Agent::Ticket::ViewForward > StateType - to define the available next states in the Forward screen.

Ticket -> Frontend::Agent::Ticket::ViewForward > StateDefault - to define the default next state of a ticket in the free text screen.

Ticket -> Frontend::Agent::Ticket::ViewForward > StateType - to define the available next states in the free text screen.

Ticket -> Core::PostMaster > PostmasterDefaultState - to define the state of tickets created from emails.

Ticket -> Core::PostMaster > PostmasterFollowUpState - to define the state of tickets after a follow-up has been received.

Ticket -> Core::PostMaster > PostmasterFollowUpStateClosed - to define the state of tickets after a follow-up has been received on an already closed ticket.

Ticket -> Core::Ticket > ViewableStateType - to define the state types that are displayed at various places in the system, for example in the Queueview.

Ticket -> Core::Ticket > UnlockStateType - to define the state types for unlocked tickets.

Ticket -> Core::Ticket > PendingReminderStateType - to define the state type for reminder tickets.

Ticket -> Core::Ticket > PendingAutoStateType - to define the state type for Pending Auto tickets.

Ticket -> Core::Ticket > StateAfterPending - to define the state a ticket is set to after the Pending Auto timer of the configured state has expired.

6.2. Ticket Priorities

OTRS comes with five default priority levels that can be modified via the "Priorities" link on the Admin page. When creating a customized list of priorities, please keep in mind that they are sorted alphabetically in the priority selection box in the user interface. Also, OTRS orders tickets by internal database IDs in the QueueView.

Note

As with other OTRS entities, priorities may not be deleted, only deactivated by setting the Valid option to *invalid* or *invalid-temporarily*.

Important

If a new priority was added or if an existing one was changed, you might also want to modify some values in SysConfig:

- Ticket:Core::Postmaster::PostmasterDefaultPriority - defines the default priority for all incoming emails.
- Ticket:Frontend::Agent:Ticket::ViewPhoneNew:Priority - defines the default priority in the New Phone Ticket screen for agents.
- Ticket:Frontend::Agent:Ticket::ViewEmailNew:Priority - defines the default priority in the New Email Ticket screen for agents.
- Ticket:Frontend::Customer:Ticket::ViewNew:PriorityDefault - defines the default priority in the New Ticket screen in the Customer frontend.

6.3. Ticket Responsibility & Ticket Watching

From OTRS 2.1 on, it is possible to assign a person as being responsible for a ticket, in addition to its owner. Moreover, all activities connected with the ticket can be watched by someone other than the ticket owner. These two functionalities are implemented with the TicketResponsible and TicketWatcher features, and facilitate the assignment of tasks and working within hierarchical team structures.

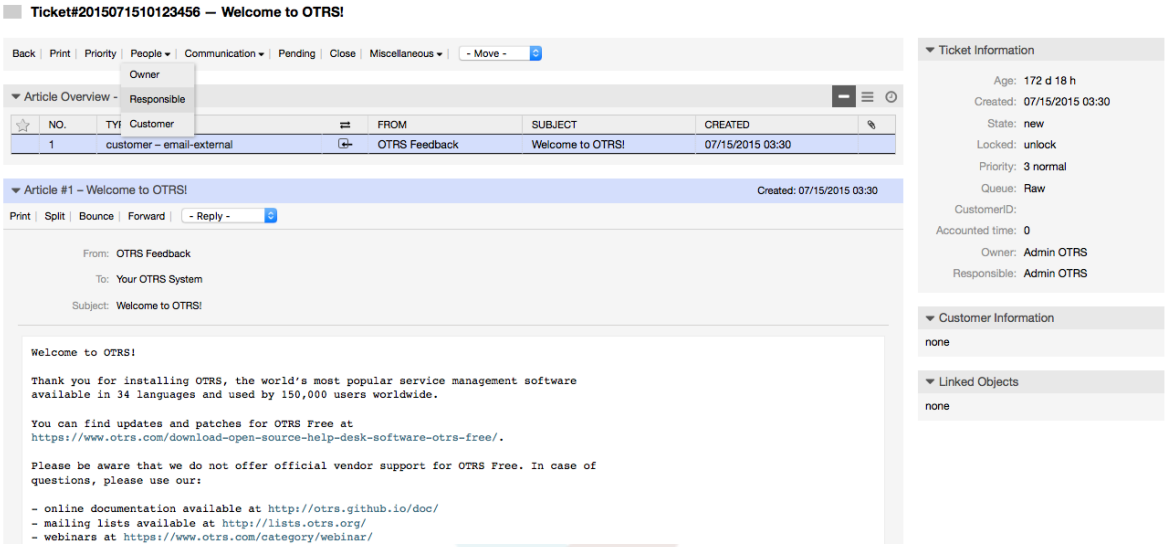
6.3.1. Ticket Responsibility

The ticket responsibility feature facilitates the complete processing of a ticket by an agent other than the ticket owner. Thus an agent who has locked a ticket can pass it on to another agent, who is not the ticket owner, in order for the second to respond to a customer request. After the request has been dealt with, the first agent can withdraw the ticket responsibility from the second agent.

With the configuration parameter Ticket::Responsible, the ticket responsibility feature can be activated. This will cause 3 new links to appear in the ticket activities menu of a zoomed ticket in the agent interface.

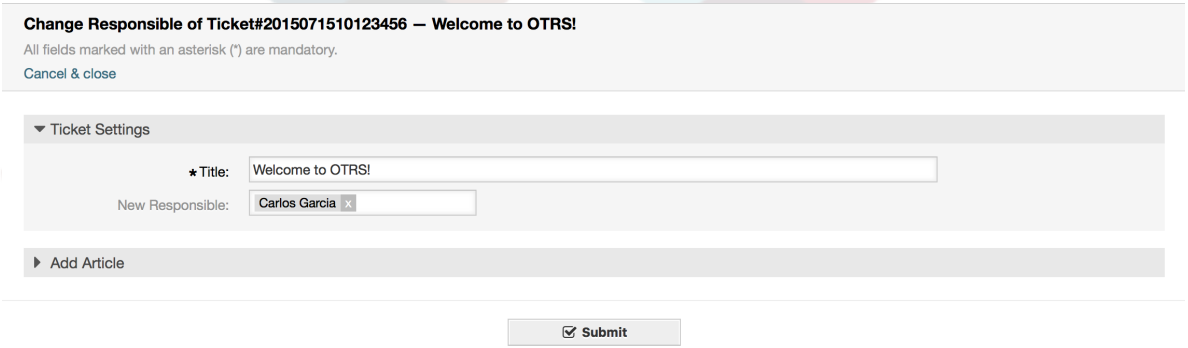
Ticket responsibility can be assigned by calling up the ticket content and clicking on the "Responsible" link in the ticket activities menu of a zoomed ticket in the agent interface (see the Figure below).

Figure 4.67. Changing the Responsibility of a ticket in its zoomed view



After clicking on "Responsible", a pop-up dialog to change the responsibility of that ticket will open (see figure below). This dialog can also be used to send a message to the new responsible agent.

Figure 4.68. Pop-up dialog to change a ticket's responsibility



The list of all tickets for which an agent is responsible, can be accessed through the Responsible view of the OTRS agent interface, as soon as the ticket responsibility feature gets activated.

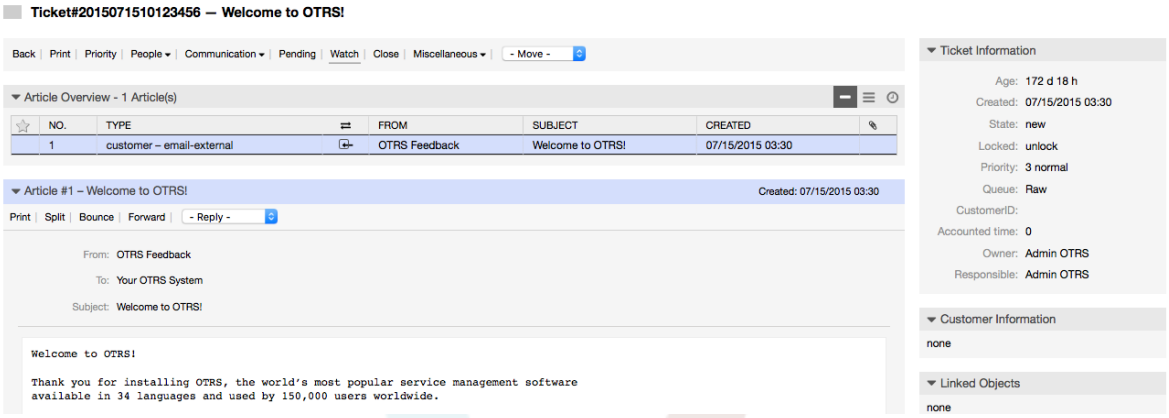
6.3.2. Ticket watching

From OTRS 2.1 on, select agents such as supervisors can watch certain tickets within the system without processing them, by using the TicketWatcher feature.

The TicketWatcher feature can be activated with the configuration parameter Ticket::Watcher which adds new links to your actions toolbar. Using Ticket::WatcherGroup, one or more user groups with permission to watch tickets can also be defined.

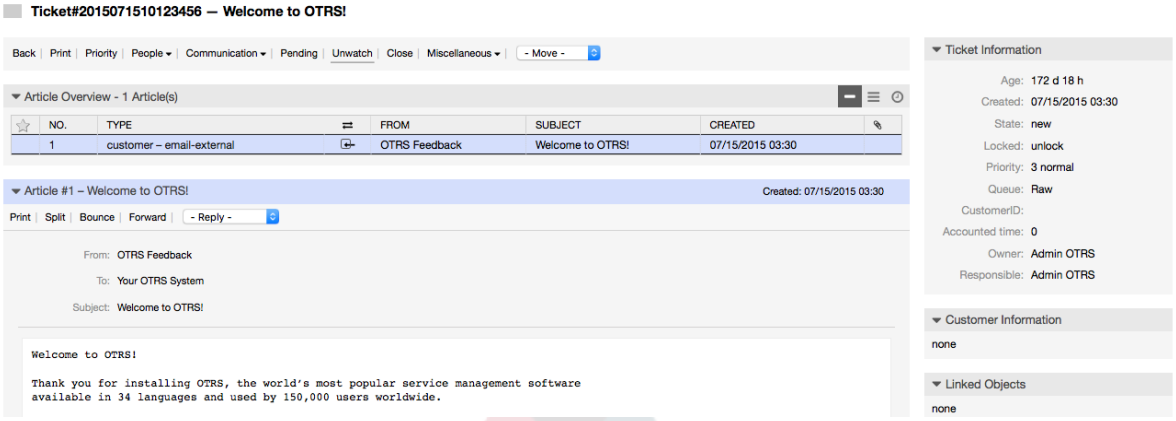
In order to watch a ticket, go to its zoomed view and click on the "Subscribe" link in the ticket activities menu (see figure below).

Figure 4.69. Subscribing to watching a ticket in its zoomed view



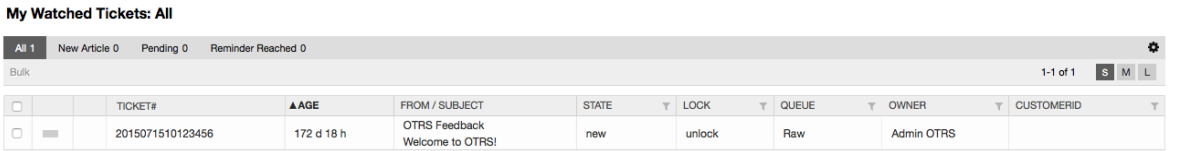
If you no longer want to watch a specific ticket, go to its zoomed view and click on the "Unsubscribe" link in the ticket activities menu (see figure below).

Figure 4.70. Unsubscribing from watching a ticket in its zoomed view



The list of all watched tickets can be accessed through the Watched view of the OTRS agent interface (see figure below), as soon as the ticket watcher feature gets activated.

Figure 4.71. Watched tickets view



7. Time Related Functions

7.1. Setting up business hours, holidays and time zones

Some functions in OTRS, like escalations and automatic unlocking of tickets, depend on a proper configuration of business hours, time zones and holidays. You can define these via the SysConfig interface, in Framework > Core::Time. You can also specify different

sets of business hours, holidays and time zones as separate 'Calendars' in Framework > Core::Time::Calendar1 through Framework > Core::Time::Calendar9. Calendars can be defined by queue settings, or on SLA levels. This means that, for example, you can specify a calendar with 5 x 8 business hours for your 'standard' SLA, but create a separate calendar with 7 x 24 support for your 'gold' SLA; as well as set a calendar for your 'Support-USA' queue with a different time window than your 'Support-Japan' queue. OTRS can handle up to 99 different calendars.

7.1.1. Business Hours

Set up the working hours for your system in SysConfig Framework > Core::Time::TimeWorkingHours, or for your specific calendar in the calendar's configuration. OTRS can handle a granularity of one hour. Checking the marks in the boxes 8, 9, 10 ... 17 corresponds with business hours of 8:00 AM - 6:00 PM.

Only during business hours can tickets escalate, notifications for escalated and pending tickets be sent, and tickets be unlocked.

7.1.2. Fixed Date Holidays

Holidays that are on a fixed date every year, such as New Year's Day or the Fourth of July, can be specified in TimeVacationDays, or in the corresponding section for the calendars 1-9.

Tickets will not escalate nor get unlocked on dates defined in TimeVacationDays.

Note

By default, OTRS ships with the *German* holidays installed.

7.1.3. Floating Holidays

Holidays such as Easter that do not have a yearly fixed date but instead vary each year, can be specified in TimeVacationDaysOneTime.

Tickets will not escalate and will not be unlocked on dates defined in TimeVacationDaysOneTime.

Note

OTRS does not ship with any One-Time holidays pre-installed. This means that you need to add holidays, such as Easter or Thanksgiving, to the system when configuring OTRS.

7.2. Automated Unlocking

Locked tickets can be automatically unlocked by the system. This feature might be useful if, for example, an agent has locked tickets that need to be processed, but he can't work on them for some reason, say because he is out of the office on an emergency. The automated unlock feature unlocks tickets after a given time to ensure that no locked tickets will be forgotten, thereby allowing other agents to process them.

The amount of time before a ticket is unlocked can be specified in the queue settings for every queue. The command `bin/otrs.Console.pl Maint::Ticket::Unlock`, which is executed periodically as a cron job, performs the automated unlocking of tickets.

Notifications on unlocked tickets are sent out only to those agents that have the queue with the unlocked tickets set in "My queues", and that have activated the notification on unlocked tickets in their personal preferences.

Tickets will be unlocked if all of the following conditions are met:

- There is an *unlock timeout* defined for the queue the ticket is in.
- The ticket is set to *locked*.
- The ticket state is *open*.

The unlock timer will be reset if an agent adds a new external article to the ticket. It can be of any of the following types: *email-external*, *phone*, *fax*, *sms*, or *note-external*.

Also, if the last article in the ticket is created by an agent, and a customer adds another one, either via web or email response, the unlock timer will be reset.

The last event that will reset the unlock timer is when the ticket is assigned to another agent.

8. Customizing the PDF Output

This section handles the configurable options for PDF output in OTRS.

If you use the Print action from anywhere within the OTRS interface, it will generate a formatted PDF file.

You can adjust the look of the files generated by OTRS by creating your own logo and adding it to PDF::LogoFile. You can use PDF::PageSize to define the standard page size of the generated PDF file (DIN-A4 or Letter), and also PDF::MaxPage to specify the maximum number of pages for a PDF file, which is useful if a user generates a huge output file by mistake.

9. Statistics

The OTRS statistics module holds features to track operational statistics and generates custom reports associated with OTRS usage. The OTRS system uses the term "statistic" generically to refer to a single report presenting various indicators.

Note

For **OTRS Business Solution™** customers, there is also a reports generator available. Here "report" refers to a collection of several statistics in one PDF document that can be easily configured and automatically generated and distributed. Please find more details in the **OTRS Business Solution™** manual.

Proper configuration of the OTRS statistics module is associated with a multitude of requirements and considerations. These include the various OTRS modules to be evaluated, user permission settings, indicators to be calculated and their complexity levels, ease of configuration of the statistics module, speed and efficiency of calculations, and support of a rich set of output variants.

Statistical elements, i.e. files which supplement the functionality of the statistics module for specific requirements, can be integrated for calculating complex statistics.

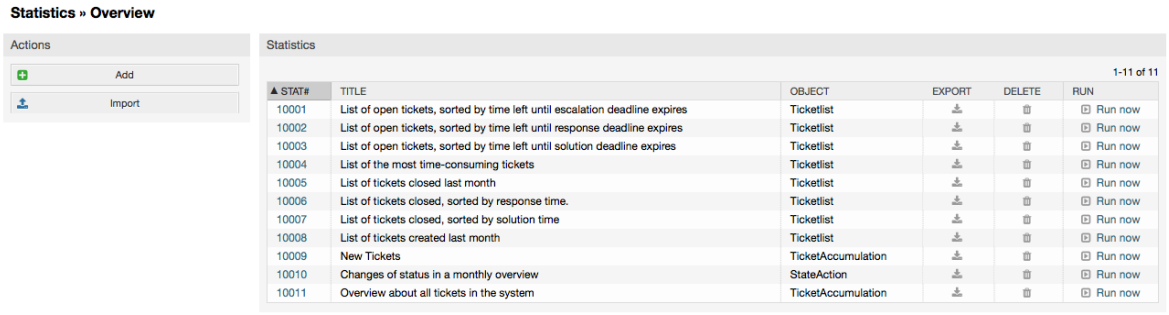
9.1. Statistics Configuration and Usage

When signed on as an agent, the statistics module can be opened by selecting "Reports" and then "Statistics" in the main menu.

9.1.1. Overview

Selecting the "Statistics" link in the navigation bar, and then the submenu link "Overview", calls up the Overview screen. The Overview screen presents a list of all pre-configured reports the agent can use (see figure below).

Figure 4.72. Overview of the standard statistics.

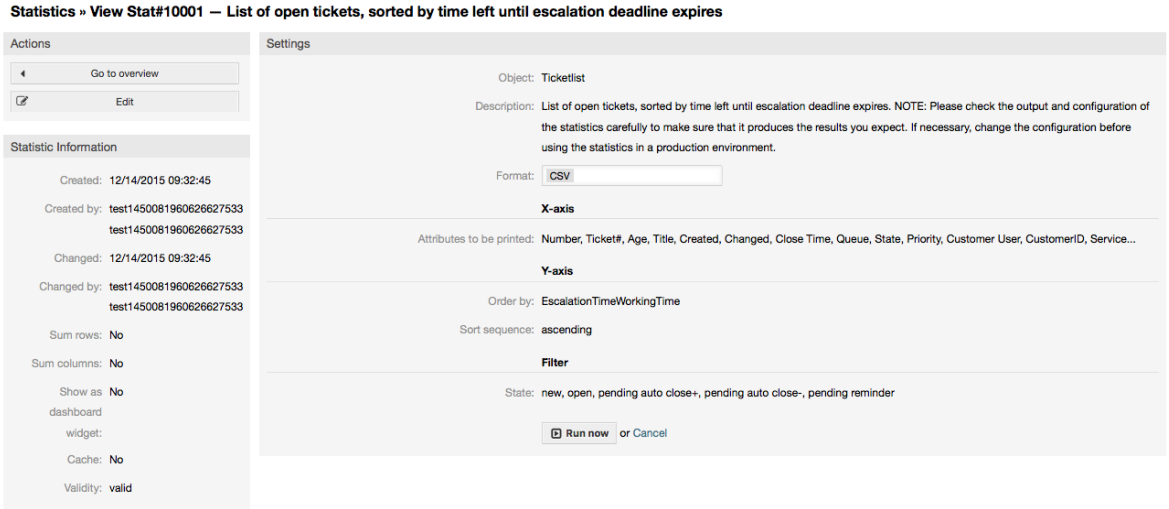


When the statistics module is installed, it comes preloaded with a few sample statistics imported into the system. These are shown as a list on the overview page. If the overview list extends to more than a single page, the agent can browse through the different pages. The list of statistics can be sorted as desired, by clicking the desired column header in the list. To generate a particular statistic, click on the statistic's "Run now" link.

9.1.2. Generation

The view user interface provides the stat's configuration settings (see figure below).

Figure 4.73. Viewing a specific statistic.



Configuration settings for a particular statistic can be set within the range of options in the View screen. Either the statistic creator or any others with the appropriate permissions can make the settings.

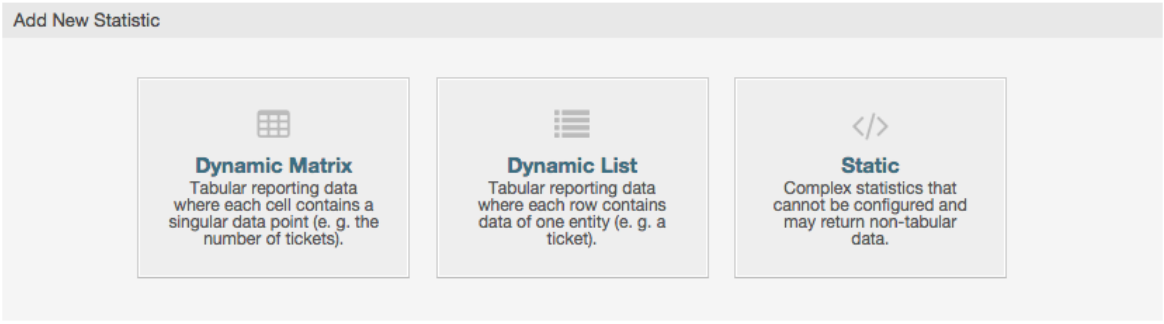
Pressing the "Start" button (at the bottom of the screen) is the last step to generate the statistic.

9.1.3. Configuration

Agents with write rights can edit an existing report configuration by calling up the edit user interface of the statistics module. Alternately, they may create a new report.

There are four possible steps in the configuration of a statistic: the general specification data, configuring the x-axis, y-axis and possible data filters for the reported data (or restrictions). Let's create a new statistic as an example by clicking the "Add" button in the overview screen. Our goal will be to get an overview of how many tickets with very high priority are in every queue (x-axis) and state (y-axis).

Figure 4.74. Adding a new statistic, first step.



At the beginning we have to select the type of statistic we want to add. Three types are available:

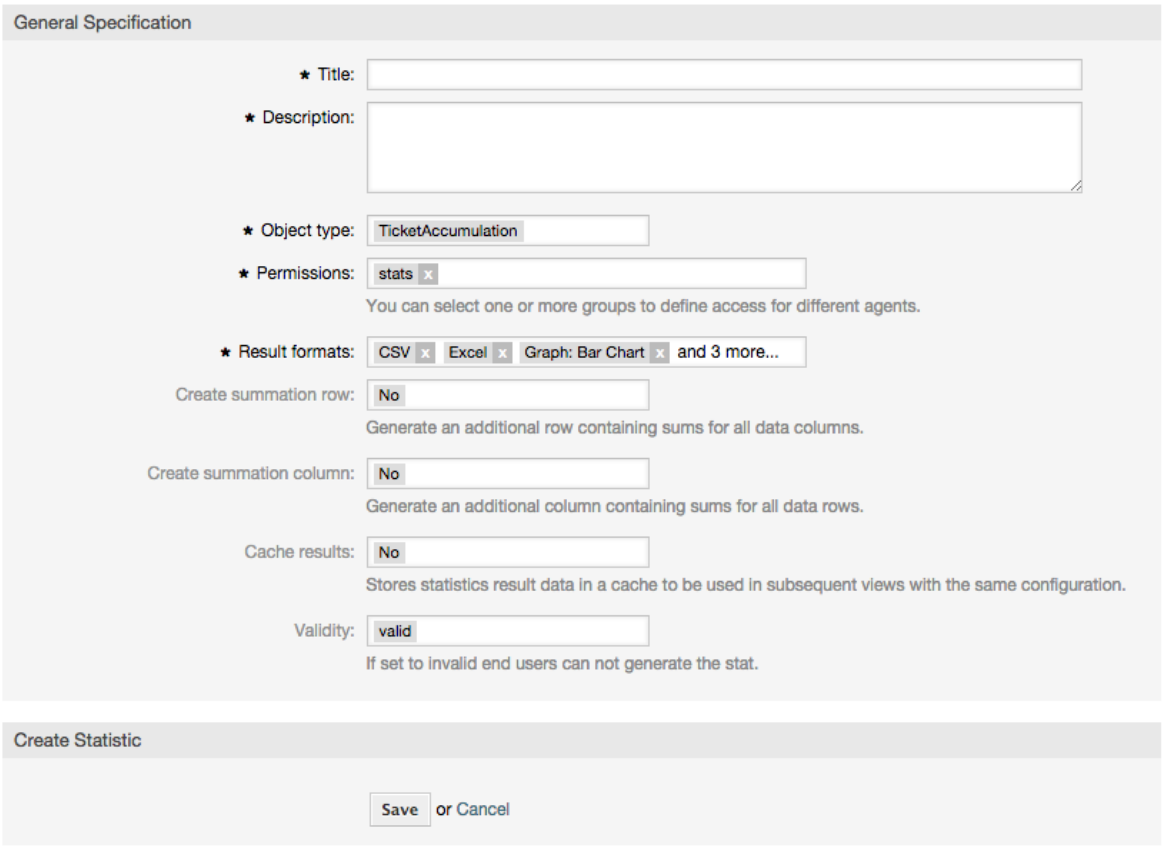
Dynamic Matrix Statistics
: This type of statistics will generate a matrix of computed values (e.g. new tickets per day of month and queue). All value cells in the matrix have the same type (number, average time, etc.). Values are computed from entities in the system (e.g. tickets). Some matrix statistics support a summation column and/or row (only useful for certain data).

Dynamic List Statistics
: This kind of statistic will generate a table where every line (not cell) represents an entity in the system (e. g. a ticket). The columns in this row are usually configurable (x-axis, see below) and contain the data of this object (e. g. ticket attributes). All value cells in one column have the same type.

Static Statistics
: This kind of statistic is not very much configurable and usually used for very special and/or complex computations.

So let's select "Dynamic Matrix" for our example. Then the "General Specifications" configuration will appear below the statistic type selection.

Figure 4.75. Adding a new statistic, second step.



After providing a title and description for the new statistic, we have to select the statistics backend that we want to use. This is the actual backend module which is responsible to collect and analyze the data for our statistic. In our case we'll select "TicketAccumulation".

By configuring permission groups, we can facilitate a restriction of the groups (and therefore, agents) who can later view and generate the pre-configured statistics. Thus the various statistics can be allocated to the different departments and work groups who need them. It is possible to allocate one statistic to various groups.

Example 4.20. Default statistics permission group

The "stats" group was selected. The report is viewable for all users having at least ro rights for the "stats" group. This access is available by default.

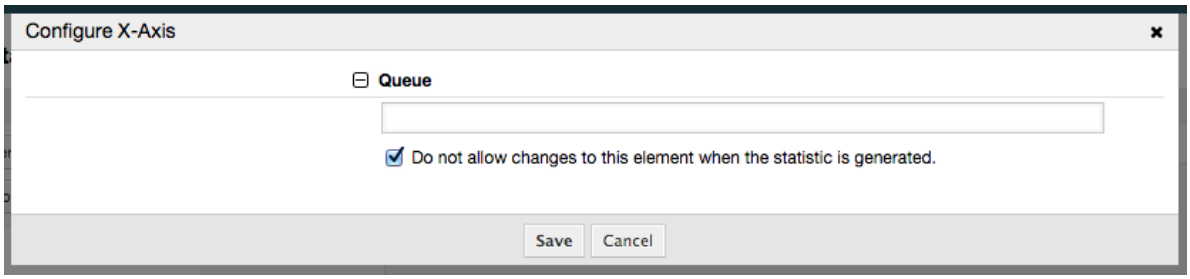
Example 4.21. Customized statistics permission group

A group named "sales" was selected. All users with ro rights for the "sales" group can see the stat in the view mode and generate it. However, the report will not be available for viewing by other users.

Additionally, possible output formats can be selected. Here we can just keep all output formats and choose the one to use when actually generating the statistic. Let's save the statistic now.

The next screen will indicate the next step with a highlighted button: we should configure the x-axis. By clicking the button, a dialog will appear where we can select the element to be used for the x-axis. In our case that will be the queue:

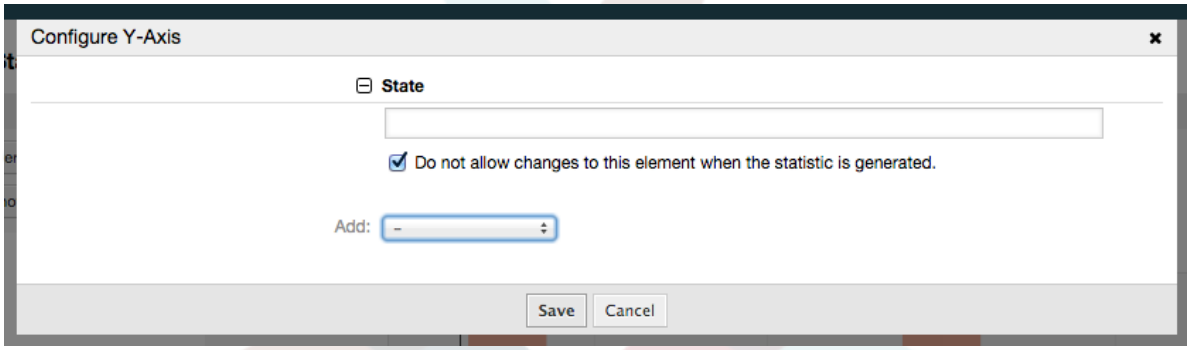
Figure 4.76. Configuring the x-axis of a statistic.



We can optionally limit the queues to be shown by selecting some in the queue field. With the checkbox we can control if the agent who generates the statistic can make changes to the queue selection. We'll keep the defaults and press "Save".

Now we can configure the y-axis in the same way: select the state field.

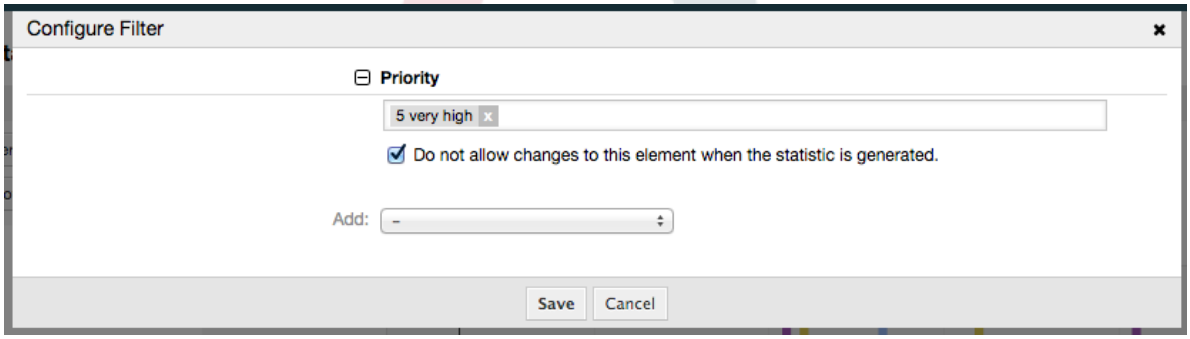
Figure 4.77. Configuring the y-axis of a statistic.



Here it is possible to select one element or two. In the first case, every value of the element will be one element on the y-axis. If two elements are selected, their permutations will be the elements on the value series. For example you could select "state" and "priority", and the resulting elements will be "new - 1 very low", "new - 2 low", ... "open - 1 very low" and so on. Let's just use the state and press "Save".

Now in the last step we could add data filters to only report tickets belonging to a certain customer, with certain priorities and so on. We'll add a filter for very high priority tickets:

Figure 4.78. Configuring the data filter of a statistic.



Now press "Save" again. The configuration is finished.

You may already have noted that in the configuration dialog there is a preview area where we can check the effect of our configuration settings.

Figure 4.79. Configuring the data filter of a statistic.

Note

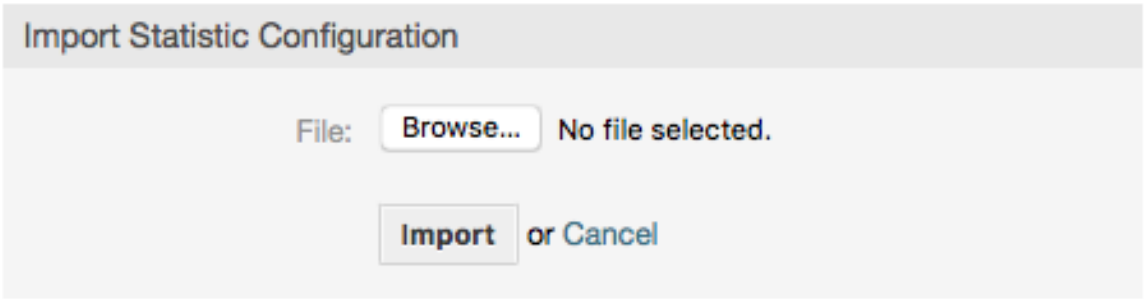
Please note that the preview uses random data and does not consider data restrictions.

The statistic is configured. By pressing the "Run now" button we can go to the View screen where the desired output format can be selected and the statistic can be generated in the different formats.

9.1.4. Import

The Import user interface can be accessed by pressing the "Import" button on the Overview screen. "rw" permissions for the statistics module are required.

Figure 4.80. Statistics import



Facilitates the import of reports, and when combined with the export function of the module, is a very handy functionality. Stats can be created and tested conveniently on test systems, then imported into the production system.

9.2. Statistics System Administration

This section provides information about the tasks and responsibilities of the OTRS administrator dealing with the statistics module.

9.2.1. Permission settings, Groups and Queues

The default configuration of the module registration gives all agents with "stats" group permissions access to the statistics module.

Access according to permission settings:

- *rw*. Allows configuring statistics.
- *ro*. Permits generating pre-configured statistics.

The OTRS administrator decides whether agents with the permission to generate pre-configured reports are allocated ro rights in the "stats" group, or if their respective groups are added in the module registration in SysConfig.

9.2.2. SysConfig Settings

The SysConfig groups Framework:Core::Stats and Framework:Frontend::Agent::Stats contain all configuration parameters for the basic set-up of the statistics module. Moreover, the configuration parameter $Self->{'Frontend::Module'}->{'AgentStats'} controls the arrangement and registration of the modules and icons within the statistics module.

9.2.3. Generating Statistics on the Command Line

Statistics can be generated on the command line with the command `bin/otrs.Console.pl Maint::Stats::Generate`. As an example, see the command line call in the following script.

```
shell> bin/otrs.Console.pl Maint::Stats::Generate --number 10004 --target-directory /tmp
Generating statistic number 10004...
  Writing file /tmp/List_of_the_most_time-consuming_tickets_Created_2015-09-08_14-51.csv.
Done.
```

A report from the statistic configuration "Stat#10004" is generated and saved as a CSV file in the `/tmp` directory.

The generated report can also be sent as an e-mail. More information can be called up with the command in the script below.

```
shell> bin/otrs.Console.pl Maint::Stats::Generate --help
```

It usually does not make sense to generate reports manually via the command line, as the statistics module has a convenient graphical user interface. However, generating reports manually does make sense when combined with a cron job.

Imagine the following scenario: On the first day of every month, the heads of department want to receive a report for the past month. By combining a cron job and command line call the reports can be sent to them automatically by e-mail.

10. Dynamic Fields

10.1. Introduction

A dynamic field is a special kind of field in OTRS, created to extend the information stored on a ticket or article. These fields are not fixed in the system and they can appear only in specific screens, they can be mandatory or not, and their representation in the screens depends on the field type defined at their creation time according to the data to be held by the field. For example, there are fields to hold a text, a date, a selection of items, etc.

Dynamic fields are the evolution of TicketFreeText, TicketFreeKey, TicketFreeTime, ArticleFreeText and ArticleFreeKey fields that where commonly used in OTRS 3.0 and before. The limitation of these "Free Fields" was that they can be defined up to 16 (text or dropdown) fields and 6 time fields for a ticket and 3 (text or dropdown) fields for each article only, not more.

Now with dynamic fields the limitation in the number of fields per ticket or article is removed, you can create as many dynamic fields you like either for ticket or articles. And beyond that, the framework behind the dynamic fields is prepared to handle custom fields for other objects rather than just ticket and articles.

This new framework that handles the dynamic fields is build using a modular approach, where each kind of dynamic field can be seen as a plug-in module for the framework. This means that the variety of dynamic fields can be easily extended by public OTRS modules, OTRS Feature Add-ons, OTRS custom developments, and other custom developments.

The following dynamic field types are included with this release:

- Text (one line of text)
- Textarea (multiple lines of text)
- Checkbox
- Dropdown (single choice, multiple values)
- Multiselect (multiple choice, multiple values)
- Date
- Date / Time

10.2. Configuration

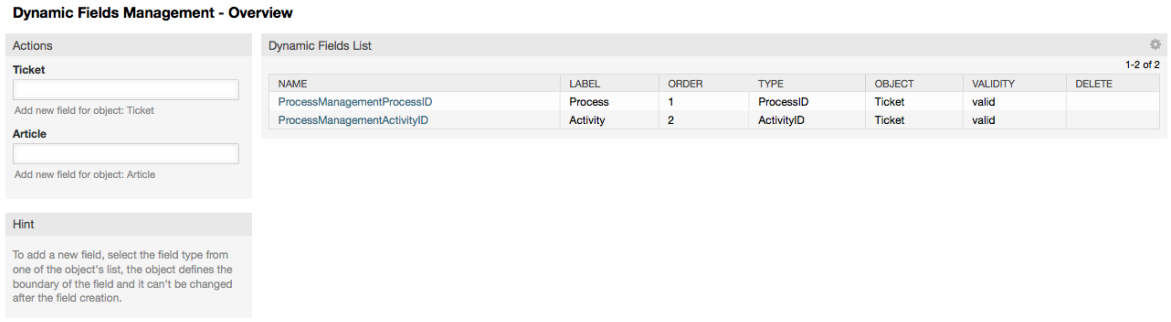
By default, a clean installation of OTRS does not include any dynamic fields. If you plan to use such fields in tickets or articles you need to create dynamic fields.

The configuration of a dynamic field is split in two parts, to add a new dynamic field or manage an existing one you need to navigate into the "Admin" panel in the "Dynamic Fields" link. To show, show as mandatory or hide a dynamic field in one screen you need to change the OTRS settings in the "SysConfig" screen.

10.2.1. Adding a Dynamic Field

Click on the "Admin" button located in the navigation bar, then click on the "Dynamic Fields" link inside "Ticket Settings" box located in the lower center of the screen. The dynamic fields overview will display as follows:

Figure 4.81. Dynamic fields overview screen, empty



Notice that this screen will change as you add more dynamic fields to list all created dynamic fields. This screen might already have some fields if the installation was updated from an older version of OTRS.

The Actions in the side bar at the left of the screen describes two possibilities: Article and Ticket, each one has it's own dropdown selection of dynamic fields.

Note

The installation of an OTRS package could add more objects to the Action side bar.

The general procedure to create a dynamic field is:

- Click on the desired dynamic field object dropdown in the Actions side bar.
- Click on the dynamic field type that you want to add from the list.
- Fill the configuration.
- Save.

The configuration dialogs for the dynamic fields are split in two parts, the upper section is common among all the fields and the lower part might be different from one type of dynamic field to another.

General dynamic field settings:

- Name: Mandatory, unique, only letters and numbers are allowed.

  This is the internal name of the field, used for example to show or hide a field in one screen. Any modification of a field name (not recommended) requires a manual update of the "SysConfig" settings where the field is referenciated.

- Label: Mandatory.

  This is the field name to be displayed on the screens, it supports translations.

  ## Note

  Label translations have to be added manually to language translations files.

- Field order: Mandatory.

  Defines the relative order in which the field will be displayed on the screen, by default each new field has the last position, a change in this setting will affect the order of the other created dynamic fields.

- Validity: Mandatory.

  An invalid dynamic field will not be displayed in any screen, no matter if is configured to displayed.

- Field type: Mandatory, Read only.

  Shows the current selected field type.

- Object type: Mandatory, Read only.

  Shows the scope of field.

Note

To illustrate each specific field type settings a few fields will be added in our example. These new fields will be referenciated in later sections.

For the following examples all the dynamic fields will be created for the Ticket object. If you need to create a dynamic field for Article object, just choose the field from the Article dropdown list.

Table 4.6. The following fields will be added into the system:

Name	Label	Type
Field1	My Field 1	Text
Field2	My Field 2	Textarea
Field3	My Field 3	Checkbox
Field4	My Field 4	Dropdown
Field5	My Field 5	Multiselect
Field6	My Field 6	Date
Field7	My Field 7	Date / Time

10.2.2. Text Dynamic Field Configuration

Text dynamic field is used to store a single line string.

Text dynamic field settings:

- Default value: Optional.

  This is the value to be shown by default on the edit screens (like New Phone Ticket or Ticket Compose).

- Show link: Optional.

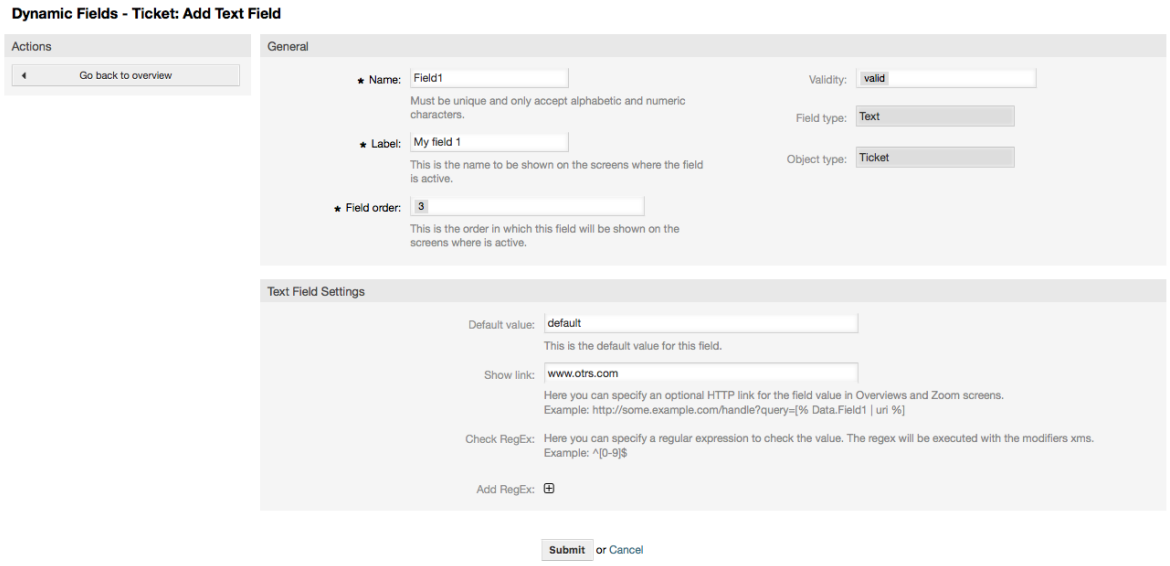
  If set, the field value will be converted into a clickable link for display screens (like ticket zoom or overviews).

  For example, if "Show link" is set to "http://www.otrs.com", clicking on the filled value will make your browser to open the OTRS web page.

Note

The use of [% Data.NameX | uri %] in the Set link value, where NameX is the name of the field, will add the field value as part of the link reference.

Figure 4.82. Dynamic field Text configuration dialog



10.2.3. Textarea Dynamic Field Configuration

Textarea dynamic field is used to store a multiple line string.

Textarea dynamic field settings:

- Number of rows: Optional, integer.

  Used to define the height of the field in the edit screens (like New Phone Ticket or Ticket Compose).

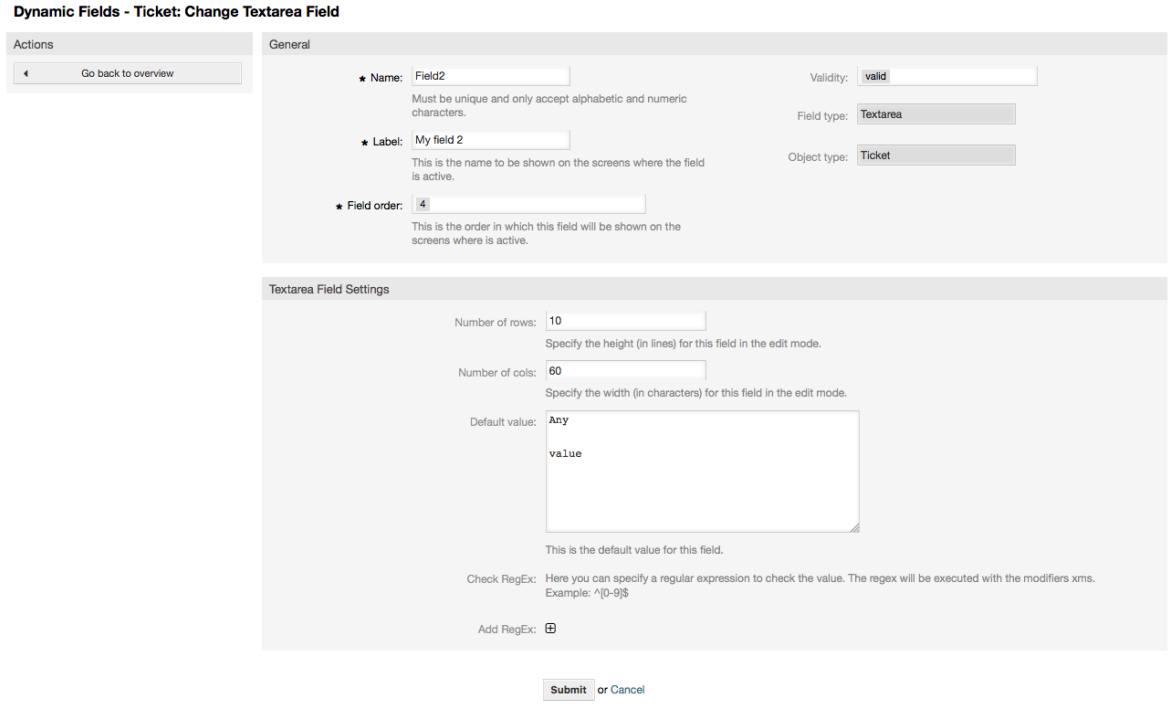
- Number of cols: Optional, Integer.

  This value is used to define the width of the field in the edit screens.

- Default value: Optional.

  This is the value to be shown by default in the edit screens (it can be a multiple line text).

Figure 4.83. Dynamic field Textarea configuration dialog



10.2.4. Checkbox Dynamic Field Configuration

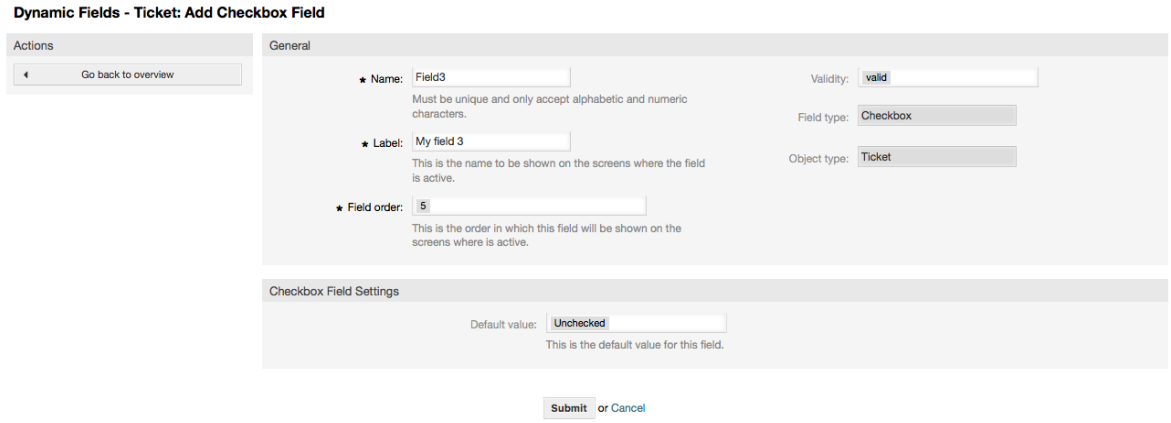
Checkbox dynamic field is used to store true or false value, represented by a checked or unchecked check box.

Checkbox dynamic field settings:

- Default value: Mandatory.

  This is the value to be shown by default on the edit screens (like New Phone Ticket or Ticket Compose), the default value for this field is closed selection which can be Checked or Unchecked.

Figure 4.84. Dynamic field Checkbox configuration dialog



10.2.5. Dropdown Dynamic Field Configuration

Dropdown dynamic field is used to store a single value, from a closed list.

Dropdown dynamic field settings:

- Possible values: Mandatory.

  List of values to choose. If used, a new value is necessary to specify the Key (internal value) and the Value (display value).

- Default value: Optional.

  This is the value to be shown by default on the edit screens (like New Phone Ticket or Ticket Compose), the default value for this field is closed selection as defined by the Possible values.

- Add empty value: Mandatory, (Yes / No).

  If this option is activated an extra value is defined to show as a "-" in the list of possible values. This special value is empty internally.

- Translatable values: Mandatory, (Yes / No).

  This setting is used mark the possible values of this field to be translated. Only the display values are translated, internal values are not affected, the translation of the values needs to be manually added to the language files.

- Show link: Optional.

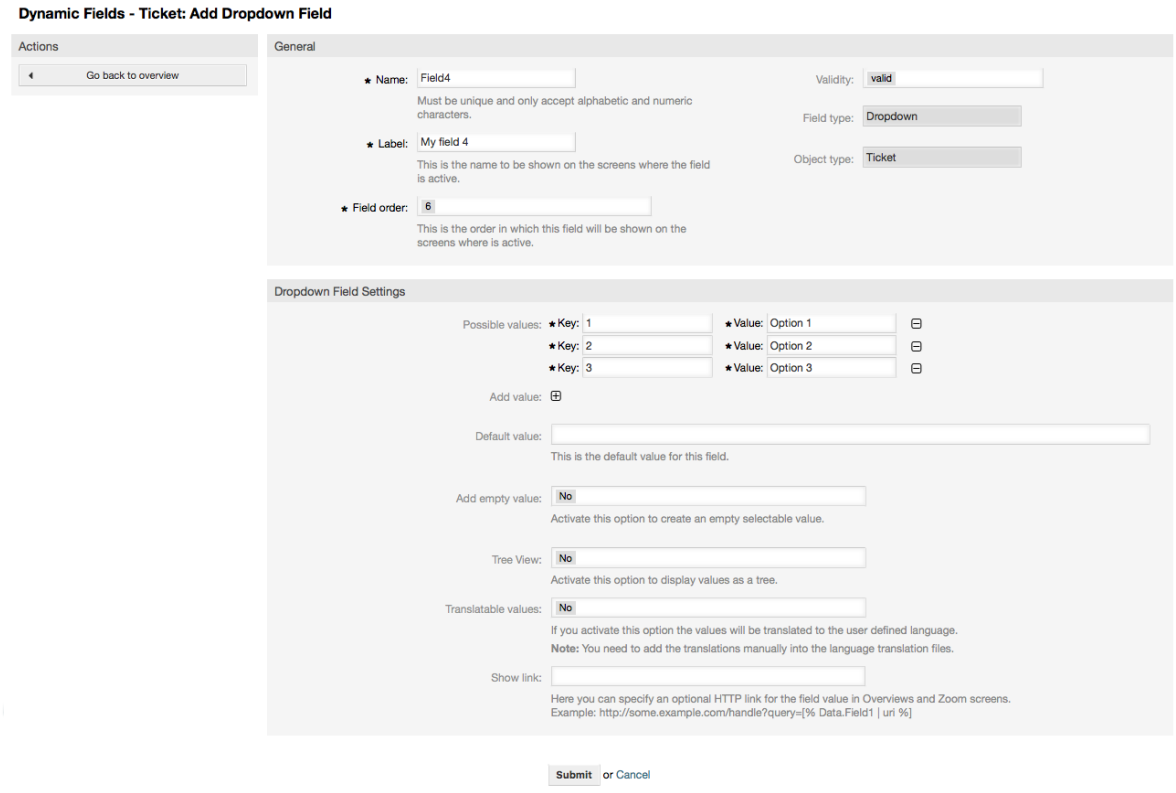
  If set, the field value will be converted into a clickable link for display screens (like ticket zoom or overviews).

  For example, if "Show link" is set to "http://www.otrs.com", clicking on the filled value will make your browser to open the OTRS web page.

  ## Note

  The use of [% Data.NameX | uri %] in the Set link value, where NameX is the name of the field, will add the field value as part of the link reference.

Figure 4.85. Dynamic field Dropdown configuration dialog



10.2.6. Multiselect Dynamic Field Configuration

Multiselect dynamic field is used to store multiple values, from a closed list.

Multiselect dynamic field settings:

- Possible values: Mandatory.

  List of values to choose from. When adding additional list items, it is necessary to specify the Key (internal value) and the Value (display value).

- Default value: Optional.

  This is the value to be shown by default on the edit screens (like New Phone Ticket or Ticket Compose), the default value for this field is closed selection as defined by the Possible values.

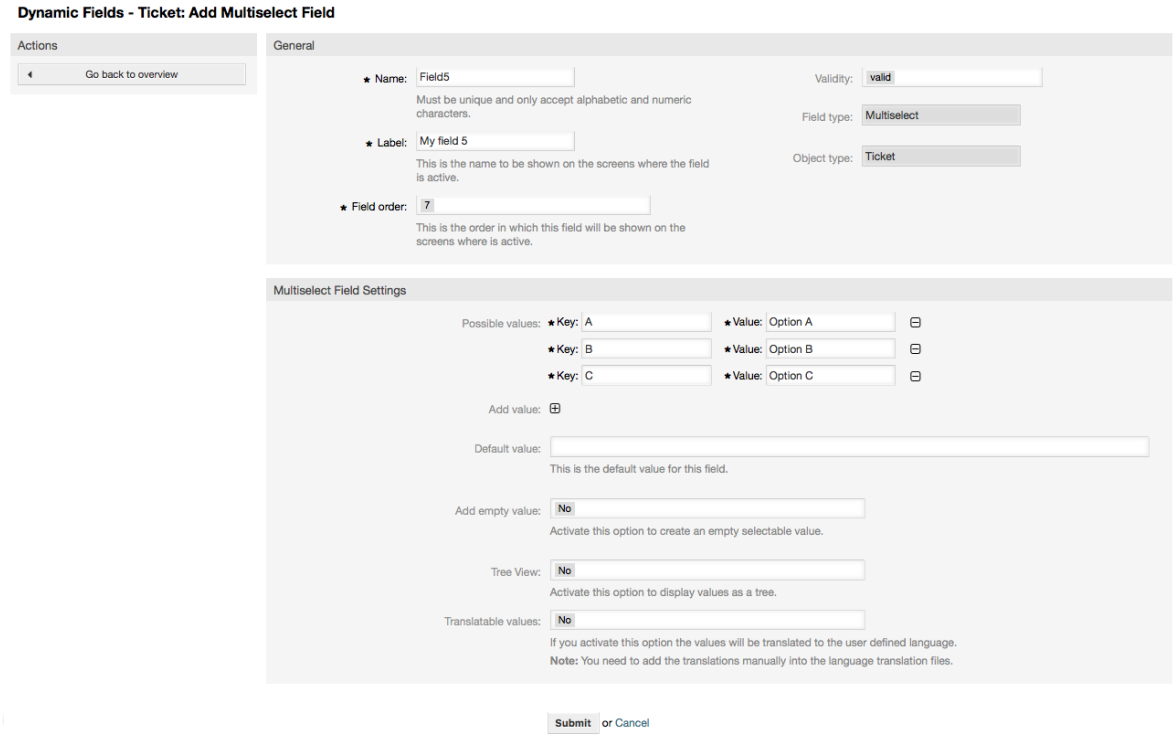
- Add empty value: Mandatory, (Yes / No).

  If this option is activated an extra value is defined to show as a "-" in the list of possible values. This special value is empty internally.

- Translatable values: Mandatory, (Yes / No).

  This setting is used mark the possible values of this field to be translated. Only the display values are translated, internal values are not affected, the translation of the values needs to be manually added to the language files.

Figure 4.86. Dynamic field Multiselect configuration dialog



10.2.7. Date Dynamic Field Configuration

Date dynamic field is used to store a date value (Day, Month and Year).

Date dynamic field settings:

- Default date difference: Optional, Integer.

  Number of seconds (positive or negative) between the current date and the selected date to be shown by default in the edit screens (like New Phone Ticket or Ticket Compose).

- Define years period: Mandatory, (Yes / No).

  Used to set a defined number of years in the past and the future based on the current date of the year select for this field. If set to Yes the following options are available:

  - Years in the past: Optional, Positive integer.

    Define the number of years in the past from the current day to display in the year selection for this dynamic field in edit screens.

  - Years in the future: Optional, Positive integer.

    Define the number of years in the future from the current day to display in the year selection for this dynamic field in edit screens.

- Show link: Optional.

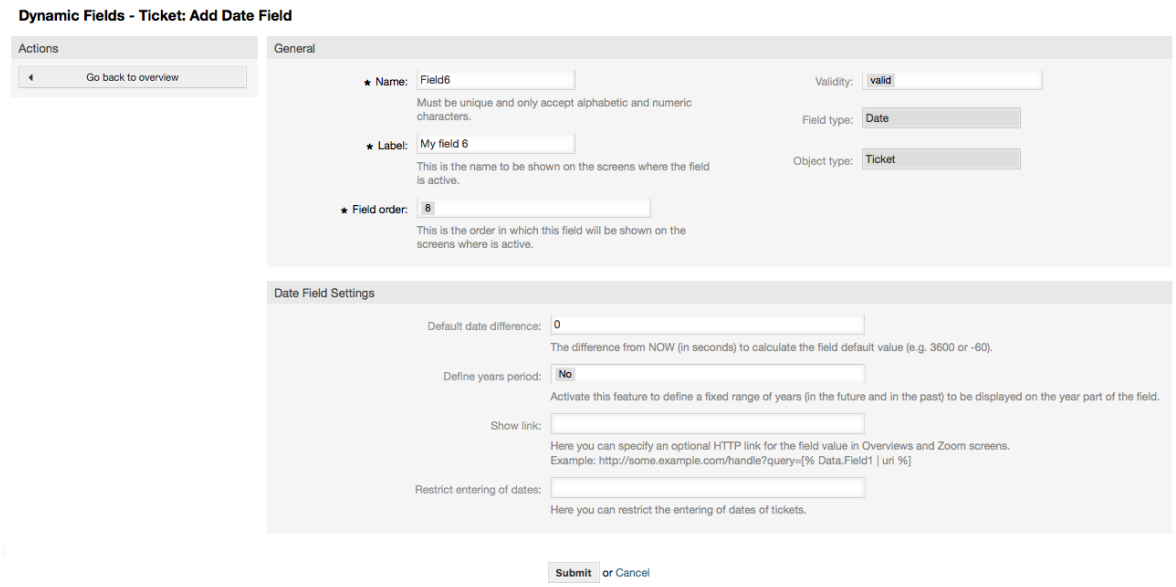
  If set, the field value will be converted into a clickable link for display screens (like ticket zoom or overviews).

  For example, if "Show link" is set to "http://www.otrs.com", clicking on the filed value will make your browser to open the OTRS web page.

Note

The use of [% Data.NameX | uri %] in the Set link value, where NameX is the name of the field will add the field value as part of the link reference.

Figure 4.87. Dynamic field Date configuration dialog



10.2.8. Date / Time Dynamic Field Configuration

Date / Time dynamic field is used to store a date time value (Minute, Hour, Day, Month and Year).

Date / Time dynamic field settings:

- Default date difference: Optional, Integer.

  Number of seconds (positive or negative) between the current date and the selected date to be shown by default in the edit screens (like New Phone Ticket or Ticket Compose).

- Define years period: Mandatory, (Yes / No).

  Used to set a defined number of years in the past and the future based on the current date of the year select for this field. If set to Yes the following options are available:

  - Years in the past: Optional, Positive integer.

    Define the number of years in the past from the current day to display in the year selection for this dynamic field in edit screens.

  - Years in the future: Optional, Positive integer.

    Define the number of years in the future from the current day to display in the year selection for this dynamic field in edit screens.

- Show link: Optional.

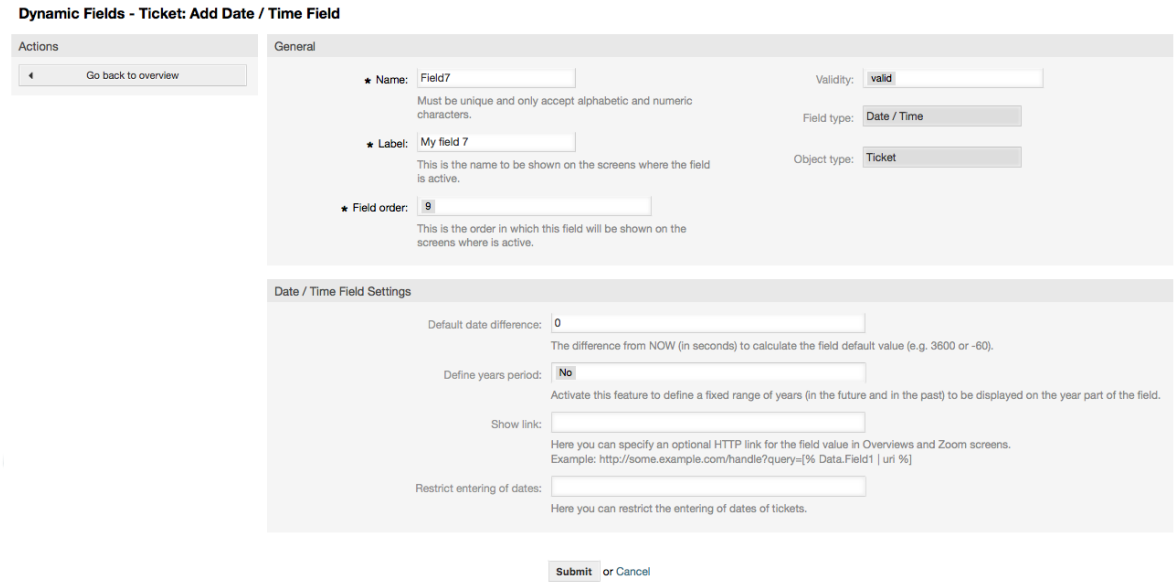
  If set, the field value will be converted into a clickable link for display screens (like ticket zoom or overviews).

For example, if "Show link" is set to "http://www.otrs.com", clicking on the filed value will make your browser to open the OTRS web page.

Note

The use of [% Data.NameX | uri %] in the Set link value, where NameX is the name of the field will add the field value as part of the link reference.

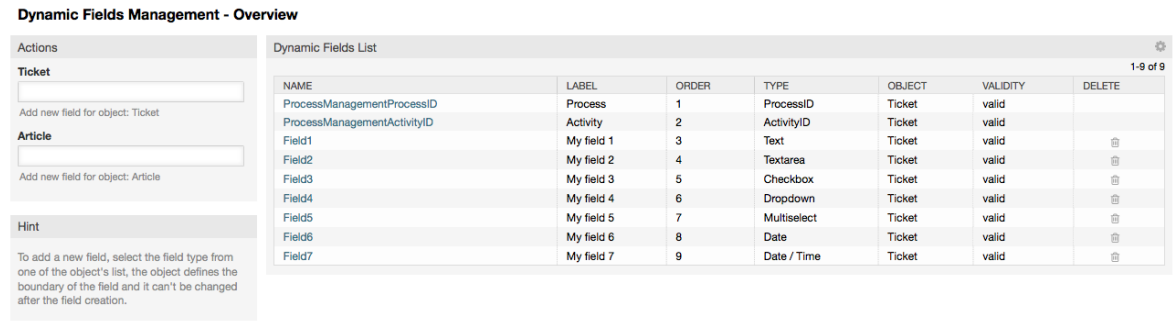
Figure 4.88. Dynamic field Date / Time configuration dialog



10.2.9. Editing a Dynamic Field

A filled dynamic field overview screen (with the previous examples) should look like:

Figure 4.89. Dynamic field overview screen filled with sample data



To change or edit a dynamic field you must have at least one field defined, select an already added field from the dynamic fields overview screen and update its settings.

Note

Not all the dynamic field settings can be changed, the Field type and Object type are fixed from the selection of the field and they can't be changed.

It is not recommended to change the field internal name, but the label can be changed at any time. If internal name is changed all "SysConfig" settings that have

a reference to that particular field needs to be updated as well as user preferences (if defined).

10.2.10. Showing a Dynamic Field on a Screen

To display a dynamic field on a particular screen there are two mandatory conditions:

1. The dynamic field must be valid.

2. The dynamic field must be set to 1 or 2 in the configuration of the screen.

Follow these steps to show a dynamic field in a screen

- Be sure that the dynamic field is set to valid, you can see the validity of the field from the dynamic field overview screen. Set to valid by editing the field if necessary.

- Open the "sysconfig" and select "Ticket" from the dropdown list in the Actions side bar located in the left part of the screen.

  ### Note

  You can also search for "DynamicField" in the search box above or the "sysconfig" key directly if you already know it.

- Locate the setting sub-group for the screen that you are looking for and click on it. For example "Frontend::Agent::Ticket::ViewPhoneNew".

- Search for the setting that ends with "###DynamicField". For example "Ticket::Frontend::AgentTicketPhone###DynamicField".

- If the setting is empty or does not have the required dynamic field name, click on the "+" button to add a new entry. For example Key: Field1, Content: 1.

  If the setting already has the dynamic field name listed be sure that is set to "1" to display the field or to "2" to display it as mandatory.

- Save the configuration by clicking on the "Update" button at the bottom of the screen and navigate to the screen where you want the field to be displayed.

10.2.10.1. Show Examples

The following are "sysconfig" configurations examples to show or hide dynamic fields on different screens.

Example 4.22. Activate Field1 in New Phone Ticket Screen.

- *Group:* Ticket

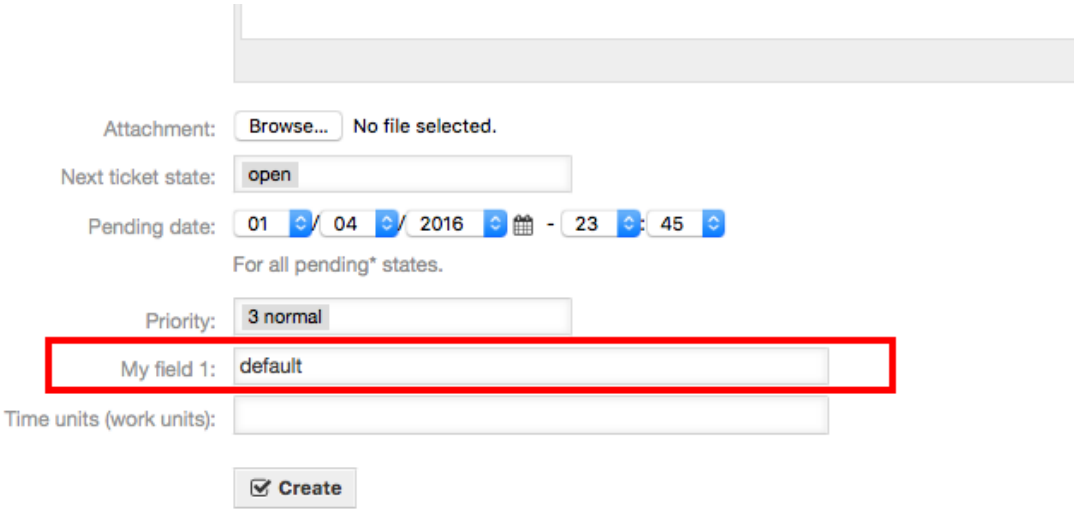
- *Sub-group:* Frontend::Agent::Ticket::ViewPhoneNew

- *Setting:* Ticket::Frontend::AgentTicketPhone###DynamicField

- *Value:*

Key	Content
Field1	1

Figure 4.90. Field1 in New Phone Ticket Screen

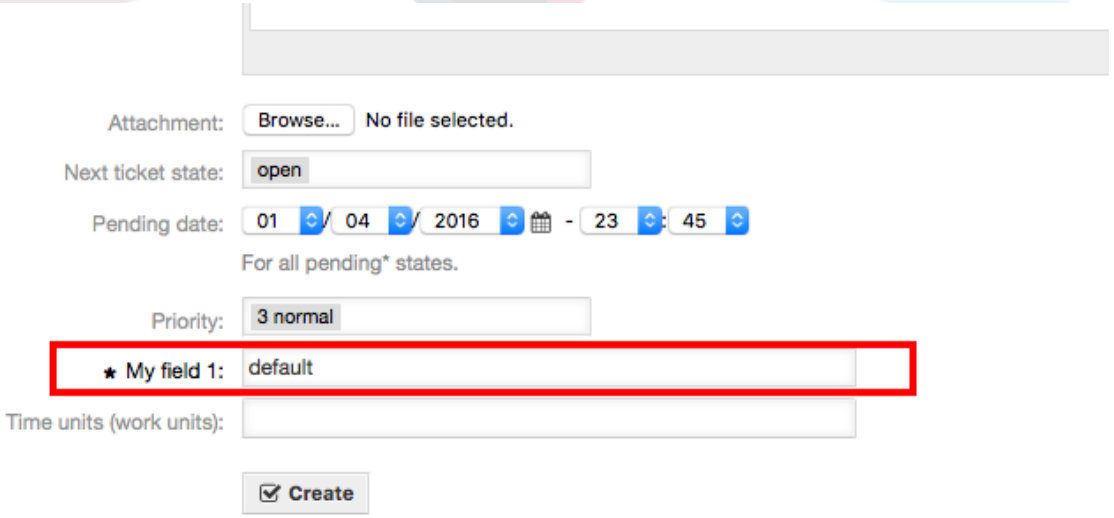


Example 4.23. Activate Field1 in New Phone Ticket Screen as mandatory.

- *Group:* Ticket
- *Sub-group:* Frontend::Agent::Ticket::ViewPhoneNew
- *Setting:* Ticket::Frontend::AgentTicketPhone###DynamicField
- *Value:*

Key	Content
Field1	2

Figure 4.91. Field1 in New Phone Ticket Screen as mandatory



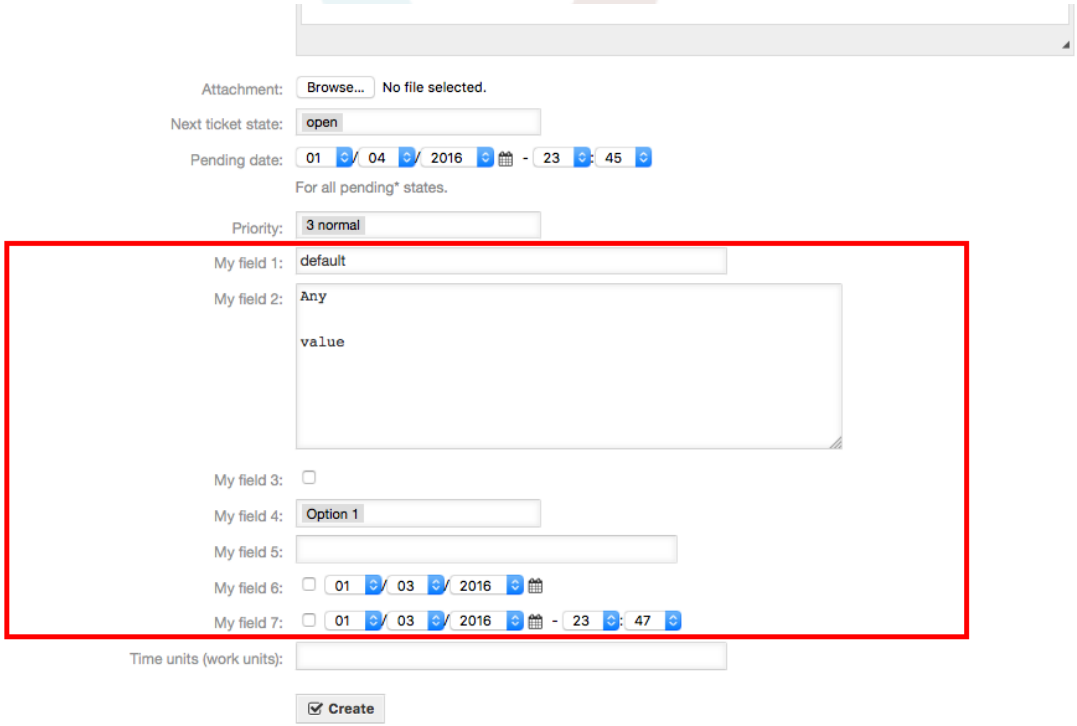
Example 4.24. Activate several fields in New Phone Ticket Screen.

- *Group:* Ticket
- *Sub-group:* Frontend::Agent::Ticket::ViewPhoneNew
- *Setting:* Ticket::Frontend::AgentTicketPhone###DynamicField

- *Value:*

Key	Content
Field1	1
Field2	1
Field3	1
Field4	1
Field5	1
Field6	1
Field7	1

Figure 4.92. Several fields in New Phone Ticket Screen as mandatory



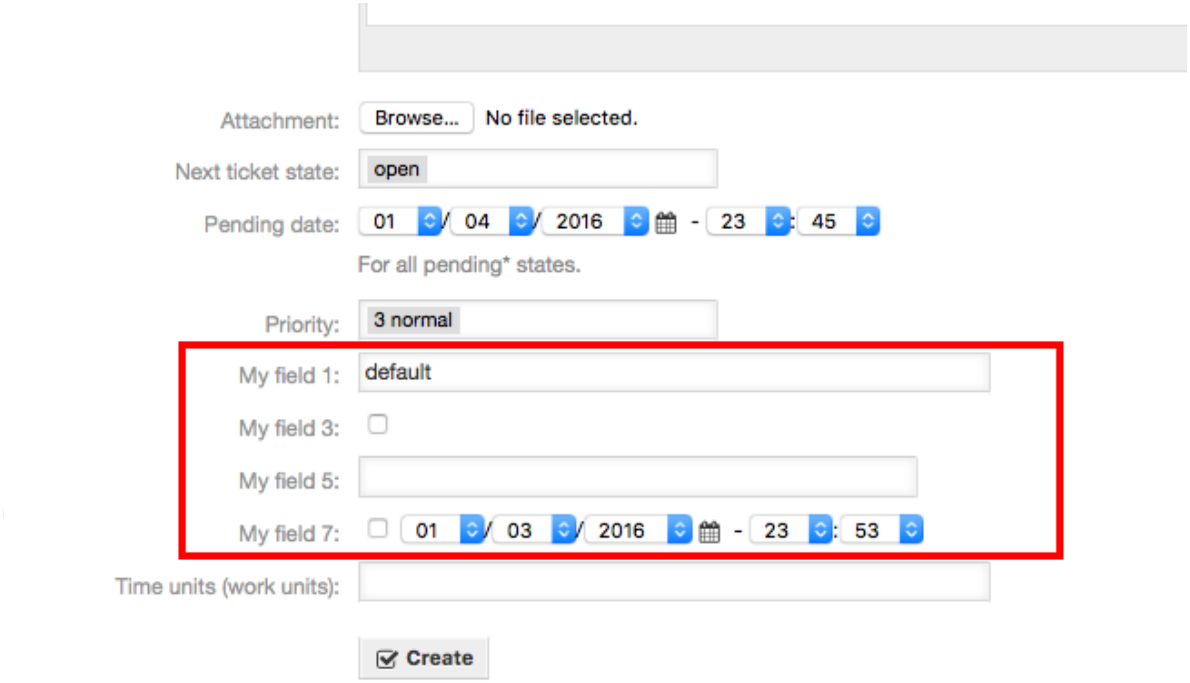
Example 4.25. Deactivate some fields in New Phone Ticket Screen.

- *Group:* Ticket
- *Sub-group:* Frontend::Agent::Ticket::ViewPhoneNew
- *Setting:* Ticket::Frontend::AgentTicketPhone###DynamicField
- *Value:*

Key	Content
Field1	1
Field2	0
Field3	1
Field4	0
Field5	1

Key	Content
Field6	0
Field7	1

Figure 4.93. Some deactivated fields in New Phone Ticket Screen as mandatory

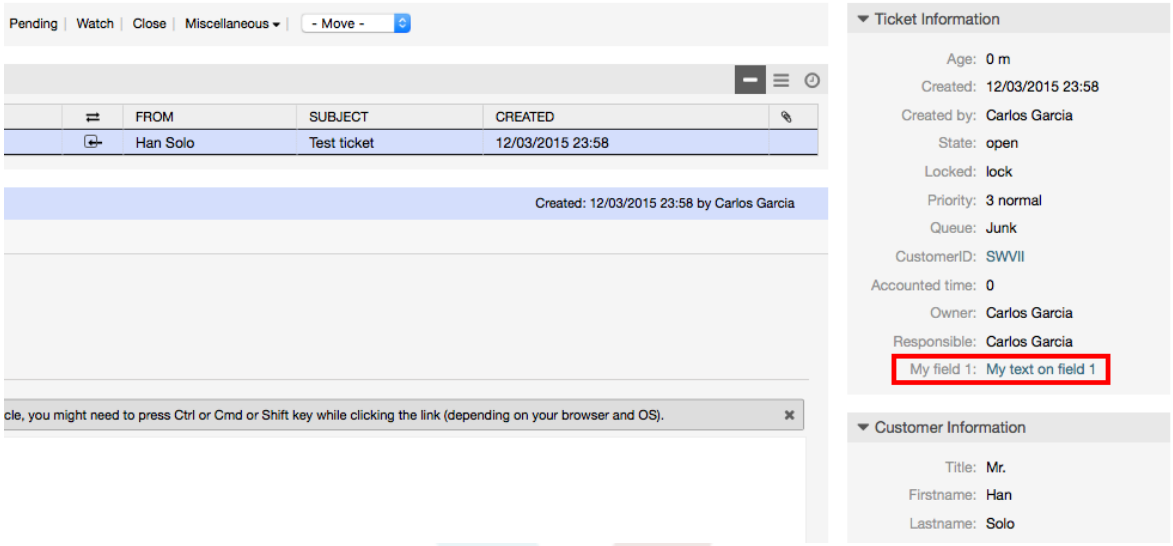


Example 4.26. Activate Field1 in Ticket Zoom Screen.

- *Group:* Ticket
- *Sub-group:* Frontend::Agent::Ticket::ViewZoom
- *Setting:* Ticket::Frontend::AgentTicketZoom###DynamicField
- *Value:*

Key	Content
Field1	1

Figure 4.94. Field1 in Ticket Zoom Screen

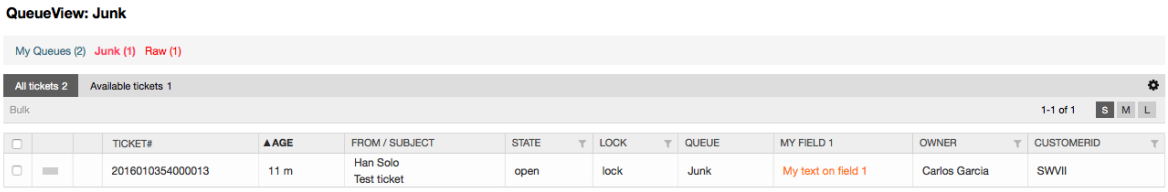


Example 4.27. Activate Field1 in Ticket Overview Small Screens.

- *Group:* Ticket
- *Sub-group:* Frontend::Agent::TicketOverview
- *Setting:* Ticket::Frontend::OverviewSmall###DynamicField
- *Value:*

Key	Content
Field1	1

Figure 4.95. Field1 in Ticket Overview Small Screen



This setting affects: Escalation View, Locked View, Queue View, Responsible View, Status View, Service View and Watch View screens.

10.2.11. Setting a Default Value by a Ticket Event Module

A ticket event (e.g. TicketCreate) can trigger a value set for a certain field, if the field does not have a value yet.

Note

By using this method this default value, is not seen in the edit screen (e.g. New Phone Ticket) since the value is set after the creation of the ticket.

To activate this feature it is necessary to enable the following setting: "Ticket::EventModulePost###TicketDynamicFieldDefault".

Example 4.28. Activate Field1 in TicketCreate event.

- *Group:* Ticket
- *Sub-group:* Core::TicketDynamicFieldDefault
- *Setting:* Ticket::TicketDynamicFieldDefault###Element1

Note

This configuration can be set in any of the 16 Ticket::TicketDynamicFieldDefault###Element settings.

If more that 16 fields needs to be set up a custom XML file must be places in $OTRS_HOME/Kernel/Config/files directory to extend this feature.

- *Value:*

Key	Content
Event	TicketCreate
Name	Field1
Value	a new value

10.2.12. Set a Default Value by User Preferences

The dynamic field default value can be overwritten with a user defined value stored in the user preferences.

Using this method, the default value of the field will be shown on any screen where the field is activated (if the field does not have already a different value).

The "sysconfig" setting "PreferencesGroups###DynamicField" located in the "Frontend::Agent::Preferences" Sub-group. This setting is an example of how to create an entry in the User Preferences screen to set an exclusive dynamic field default value for the selected user. The limitation of this setting is that it only permits the use of one dynamic field. If two or more fields will use this feature, it is necessary to create a custom XML configuration file to add more settings similar to this one.

Note

Remember, if more settings are added in a new XML each setting name needs to be unique in the system and different than "PreferencesGroups###DynamicField". For example: PreferencesGroups###101-DynamicField-Field1, PreferencesGroups###102-DynamicField-Field2, PreferencesGroups###My-Field1, PreferencesGroups###My-Field2, etc.

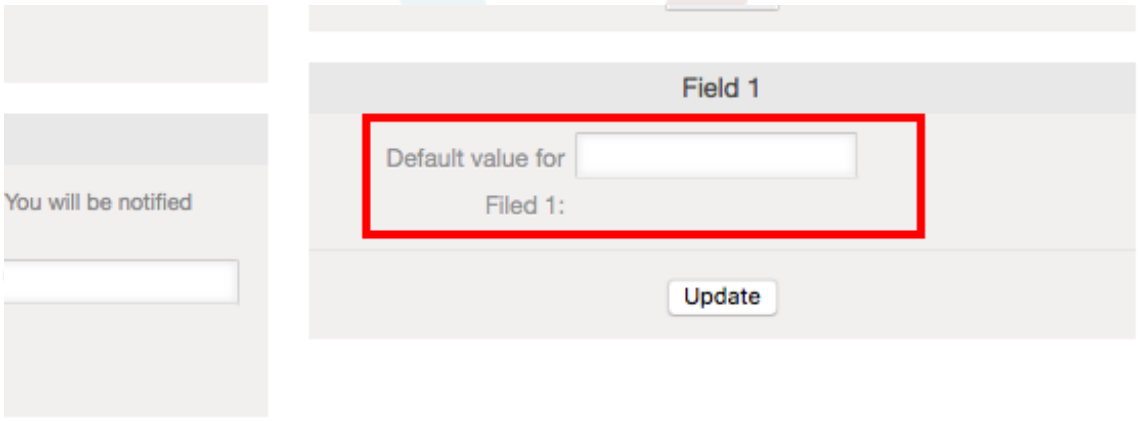
Example 4.29. Activate Field1 in the User preferences.

- *Group:* Ticket
- *Sub-group:* Frontend::Agent::Preferences
- *Setting:* PreferencesGroups###101-DynamicField-Field1
- *Value:*

Key	Content
Event	TicketCreate

Key	Content
Active	1
Block	Input
Column	Other Settings
Data:	[% Env("UserDynamicField_Field1") %]
Key:	My Field 1
Label:	Default value for: My Field 1
Module:	Kernel::Output::HTML::PreferencesGeneric
PrefKey:	UserDynamicField_Field1
Prio:	7000

Figure 4.96. Field1 in User preferences screen



11. Generic Interface

The OTRS Generic Interface consists of a multiple layer framework that lets OTRS communicate with other systems via a web service. This communication could be bi-directional:

- *OTRS as Provider:* OTRS acts as a server listening to requests from the External System, processing the information, performing the requested action, and answering the request.
- *OTRS as Requester:* OTRS acts as a client collecting information, sending the request to the Remote System, and waiting for the response.

11.1. Generic Interface Layers

Generic Interface is build based on a layer model, to be flexible and easy to customize.

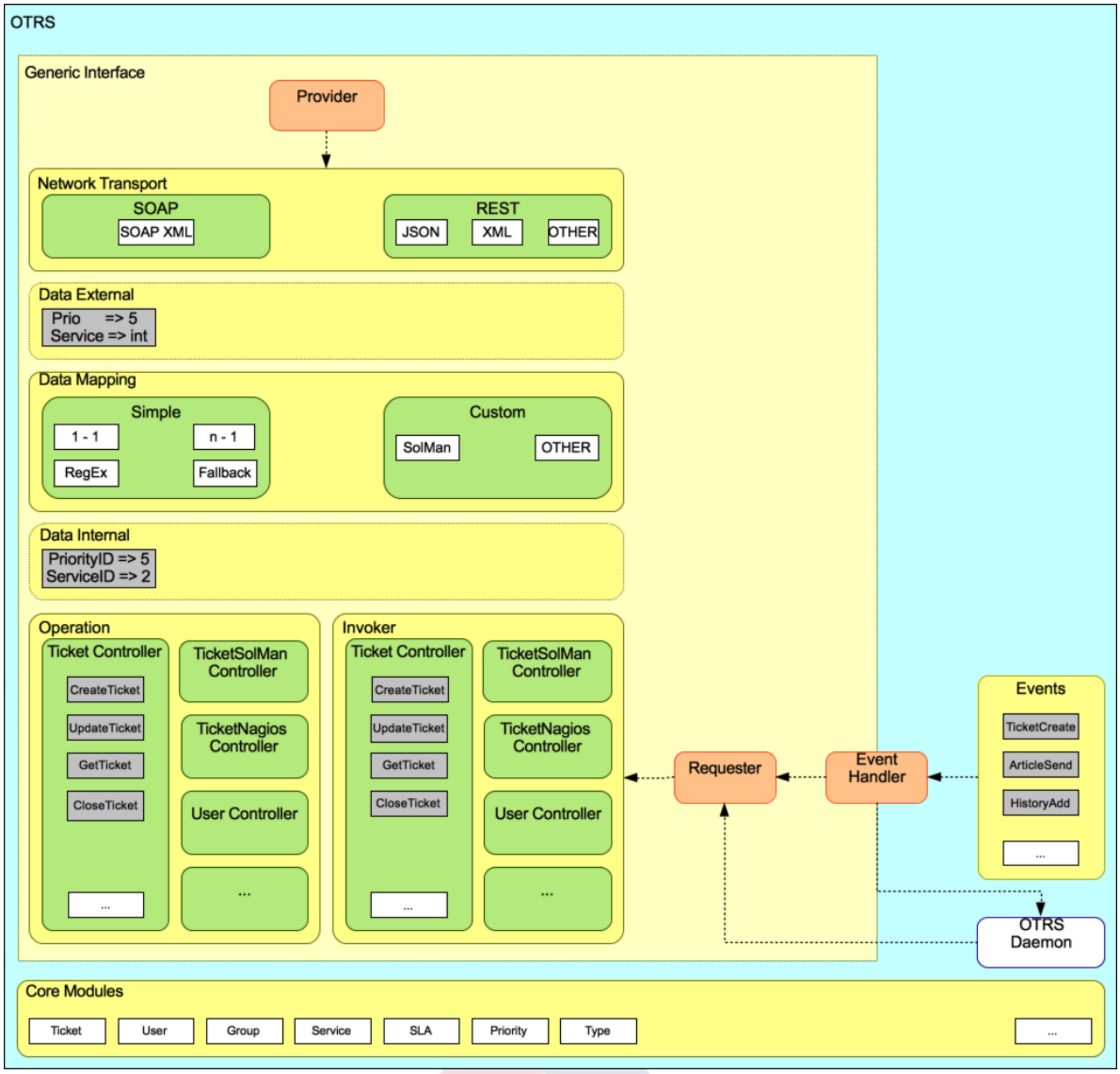
A layer is a set of files, which control how the Generic Interface performs different parts of a web service. Using the right configuration, one can build different web services for different External Systems without creating new modules.

Note

If the Remote System does not support the current bundled modules of the Generic Interface, special modules need to be developed for that specific web service.

The list of provided Generic Interface modules shipped with OTRS will be updated and increased over time.

Figure 4.97. The graphical interface layers



11.1.1. Network Transport

This layer is responsible for the correct communication with the Remote System. It receives requests and generates responses when acting as provider, and generates requests and receives responses when acting as requester.

Provider communication is handled by a new web server handle called "nph-genericinterface.pl".

Requester communication could be initiated during an event triggered by a Generic Interface module or any other OTRS module. This event is catched by the event handler and depending on the configuration the event will be processed directly by the requester object or delegated to the Scheduler (a separated daemon designed to process tasks asynchronously).

11.1.2. Data Mapping

This layer is responsible for translating data structures between OTRS and the Remote System (data internal and data external layers). Usually Remote Systems have different data structures than OTRS (including different values and names for those values), and

here resides the importance of the layer to change the received information into something that OTRS can understand and on the opposite way send the information to each Remote System using their data dictionaries.

Example: "Priority" (OTRS) might be called "Prio" in a remote system and it could be that value "1 Low" (OTRS) should be mapped to "Information" on the remote system.

11.1.3. Controller

Controllers are collections of similar Operations or Invokers. For example, a Ticket controller might contain several standard ticket operations. Custom controllers can be implemented, for example a "TicketExternalCompany" controller which may contain similar functions as the standard Ticket controller, but with a different data interface, or function names (to adapt to the Remote System function names) or complete different code.

One application for Generic Interface could be to synchronize information with one Remote System that only can talk with another Remote System of the same kind. In this case new controllers needs to be developed and the Operations and Invokers has to emulate the Remote System behavior in such way that the interface that OTRS exposes is similar to the Remote System's interface.

11.1.4. Operation (OTRS as a provider)

An Operation is a single action that can be performed within OTRS. All operations have the same programming interface, they receive the data into one specific parameter, and return a data structure with a success status, potential error message and returning data.

Normally operations uses the already mapped data (internal) to call core modules and perform actions in OTRS like: Create a Ticket, Update a User, Invalidate a Queue, Send a Notification, etc. An operation has full access to the OTRS API to perform the action.

11.1.5. Invoker (OTRS as a requester)

An Invoker is an action that OTRS performs against a Remote System. Invokers use the OTRS Core modules to process and collect the needed information to create the request. When the information is ready it has to be mapped to the Remote System format in order to be sent to the Remote System, that will process the information execute the action and send the response back, to either process the success or handle errors.

11.2. Generic Interface Communication Flow

The Generic Interface has a defined flow to perform actions as a provider and as a requester.

These flows are described below:

11.2.1. OTRS as Provider

11.2.1.1. Remote Request:

1. HTTP request
   - OTRS receives HTTP request and passes it through the layers.
   - The provider module is in charge to execute and control these actions.
2. Network Transport

- The network transport module decodes the data payload and separates the operation name from the rest of the data.
- The operation name and the operation data are returned to the provider.

3. *Data External*
   - Data as sent from the remote system (This is not a module-based layer).
4. Mapping
   - The data is transformed from the External System format to the OTRS internal format as specified in the mapping configuration for this operation (Mapping for incoming request data).
   - The already transformed data is returned to the provider.
5. *Data Internal*
   - Data as transformed and prepared to be passed to the operation (This is not a module based layer).
6. Operation
   - Receives and validates data.
   - Performs user access control.
   - Executes the action.

11.2.1.2. OTRS Response:

1. Operation
   - Returns result data to the provider.
2. *Data Internal*
   - Data as returned from operation.
3. Mapping
   - The data is transformed back to the Remote system format as specified in the mapping configuration (Mapping for outgoing response data).
   - The already transformed data is returned to the provider.
4. *Data external*
   - Data as transformed and prepared to be passed to Network Transport as response.
5. Network Transport
   - Receives the data already in the Remote System format.
   - Constructs a valid response for this network transport type.
6. HTTP response
   - The response is sent back to the web service client.

- In the case of an error, an error response is sent to the remote system (e.g. SOAP fault, HTTP error, etc).

11.2.2. OTRS as Requester

11.2.2.1. OTRS Request:

1. Event Trigger Handler
   - Based on the web service configuration determines if the request will be synchronous or asynchronous.
     - Synchronous
       - A direct call to the Requester is made in order to create a new request and to pass it through the layers.
     - Asynchronous
       - Create a new Generic Interface (Requester) task for the OTRS Daemon (by delegating the request execution to the Scehduler Daemon, the user experience could be highly improved, otherwise all the time needed to prepare the request and the remote execution will be added to the OTRS Events that trigger those requests).
       - In its next cycle the OTRS daemon process reads the new task and creates a call to the Requester that will create a new request and then passes it through the layers.
2. Invoker
   - Receives data from the event.
   - Validates received data (if needed).
   - Call core modules to complement the data (if needed).
   - Return the request data structure or send a Stop Communication signal to the requester, to gracefully cancel the request.
3. *Data Internal*
   - Data as passed from the invoker (This is not a module based layer).
4. Mapping
   - The data is transformed to the Remote system format as specified in the mapping configuration (Mapping for outgoing response data).
   - The already transformed data is returned to the requester.
5. *Data External*
   - Data as transformed and prepared for sending to the remote system.
6. Network Transport
   - Receives the remote operation name and the data already transformed to the Remote System format from the requester.
   - Constructs a valid request for the network transport.

- Sends the request to the remote system and waits for the response.

11.2.2.2. Remote Response:

1. Network transport
   - Receives the response and decodes the data payload.
   - Returns the data to the requester.
2. *Data External*
   - Data as received from the Remote System.
3. Mapping
   - The data is transformed from the External System format to the OTRS internal format as specified in the mapping configuration for this operation (Mapping for incoming response data).
   - The already transformed data is returned to the requester.
4. *Data Internal*
   - Data as transformed and ready to be passed back to the requester.
5. Invoker
   - Receives return data.
   - Handles the data as needed specifically by each Invoker (included error handling if any).
   - Return the Invoker result and data to the Requester.
6. Event Handler or OTRS Daemon
   - Receives the data from the Requester. In the case of the OTRS Daemon this data might contain information to create a task in the future.

11.3. Web Services

A Web Service is a communication method between two systems, in our case OTRS and a Remote System.

The heart of the Web Service is its configuration, where it is defined what actions the web service can perform internally (Operation), what actions the OTRS request can perform Remote System (Invokers), how data is converted from one system to the other (Mapping), and over which protocol the communication will take place (Transport).

The Generic Interface is the framework that makes it possible to create Web Services for OTRS in a predefined way, using already made building blocks that are independent from each other and interchangeable.

11.4. Web Service Graphical Interface

The web service graphical user interface (GUI) is a tool that allows to construct complex web service configurations in a user friendly and convenient interface. It allows to:

- Create and Delete web services.

- Import and Export configurations (in YAML file format) for existing web services.
- View, Revert and Export old configurations for existing web services in the Web Service History screen.
- Track all communication logs for each web service in the Debugger screen.

11.4.1. Web Service Overview

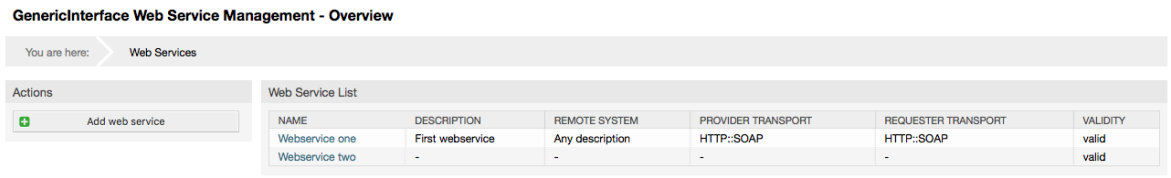
The "Web Services" link in the main screen of Admin Interface (in the System Administration box) leads to the web services overview screen, where you are able to manage your web service configurations. You can add new web services or change the configuration of the existing ones from this screen.

Every web service configuration screen has in the upper part of the screen a "bread crumbs" style navigation path. This navigation path is useful to know exactly in which part of the web service configuration we are, and also enables the user to jump back to any part of the configuration process at any time (this action will not save any changes).

Note

To create a new web service, press the button "Add web service", and provide the required information.

Figure 4.98. Web services overview



11.4.2. Web Service Add

The only required field in this part is the web service "Name" that needs to be unique in the system and can not be left empty. Other fields are also necessary for the configuration like the "Debug Threshold" and "Validity" but these fields are already populated with the default value for each list.

The default value for "Debug Threshold" is "debug". When configured in this manner all communication logs are registered in the database. Each subsequent Debug Threshold value is more restrictive and discards communication logs of lower order than the one set in the system.

Debug Threshold levels (from lower to upper)

- Debug
- Info
- Notice
- Error

It is also possible to define the network transport protocol for "OTRS as Provider" and "OTRS as requester".

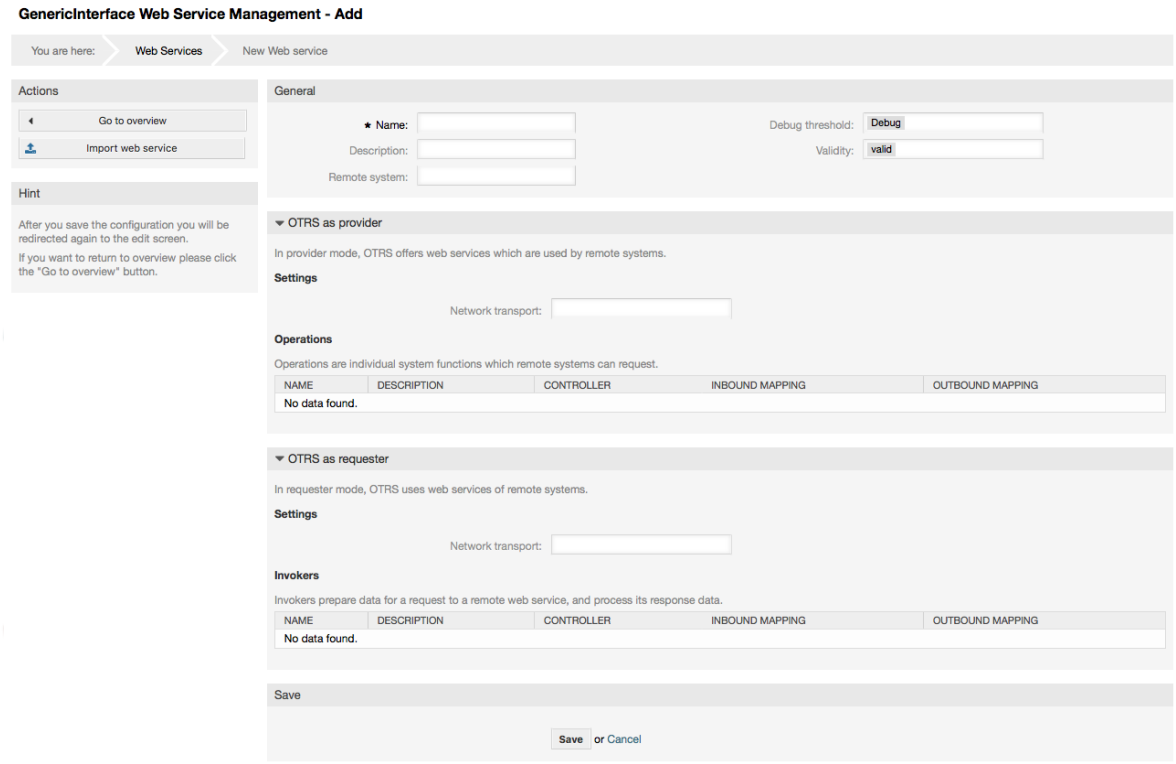
Click on the "Save" button to register the new web service in the database or click "Cancel" to discard this operation. You will now be returned to the web service overview screen.

If you already have a web service configuration file in YAML format you can click on the "Import web service" button on the left side of the screen. For more information on importing web services please check the next section "Web Service Change".

Note

To change or add more details to a web service, click on the web service name in the web service overview screen.

Figure 4.99. Web services add



11.4.3. Web Service Example Import

Did you know there are example web services available in the OTRS Business Solution™?

11.4.4. Web Service Change

On this screen you have a complete set of functions to handle every part of a web service. On the left side in the action column you can find some buttons that allows you to perform all possible actions on a web service:

- Clone web service.
- Export web service.
- Import web service.
- Configuration History.
- Delete web service.

- Debugger.

Note

"Configuration history" and "Debugger" will lead you to different screens.

11.4.4.1. Web Service Clone

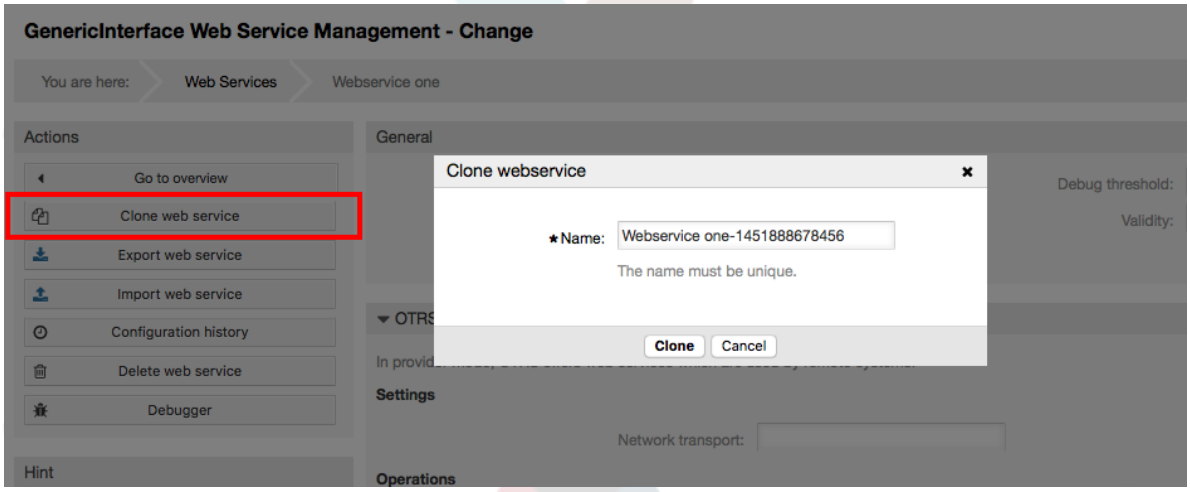
To clone a web service, you need to click on the "Clone web service" button. A dialog will be shown where you can use the default name or set a new name for the (cloned) web service.

Note

Remember that the name of the web service must be unique within the system.

Click on "Clone" button to create the web service clone or "Cancel" to close the dialog.

Figure 4.100. Web service clone



11.4.4.2. Web Service Export

The "Export web service" button gives you the opportunity to dump the configuration of the current web service into a YAML file, to download it and to store it on your file system. This can be specially useful if you want to migrate the web service from one server to another, for example from a testing environment to a production system.

Warning

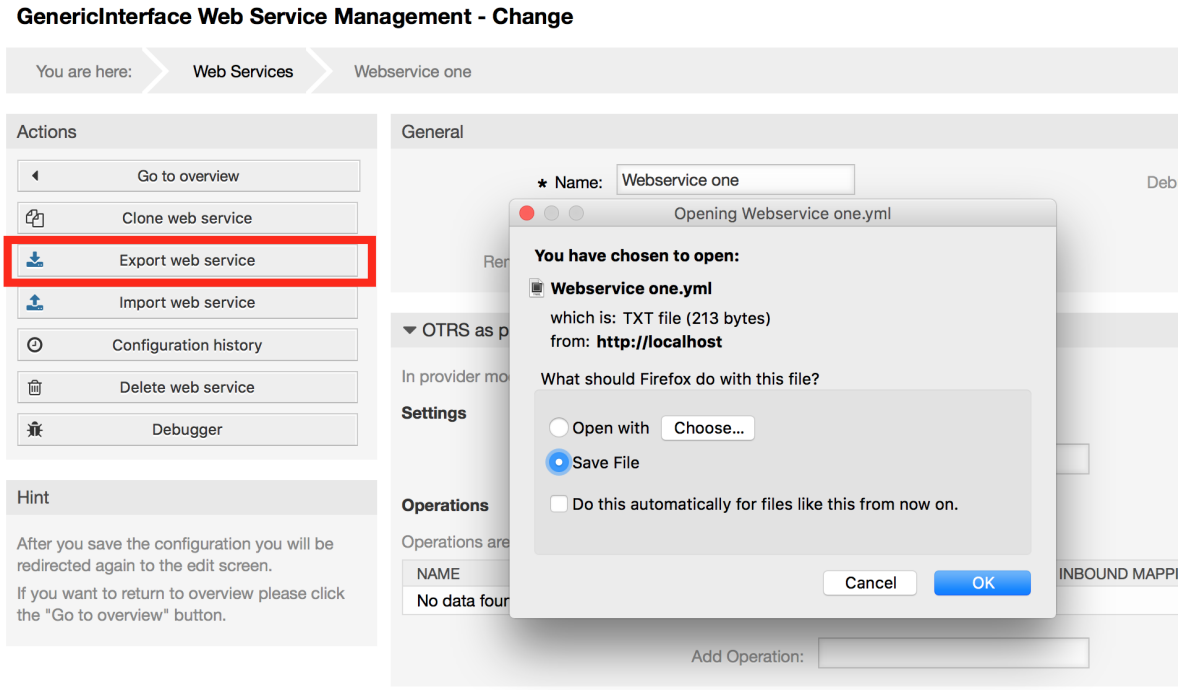
All stored passwords in the web service configuration will be exported in plain text format.

Right after clicking the "Export web service" button a save dialog of your browser will appear, just like when you click on a file download link on a web page.

Note

Each browser on each operating system has its own save dialog screen and style. Depending on the browser and its configuration it is possible that no dialog is shown and the file is saved to a default directory on your file system. Please check your browser documentation for more specific instructions if needed.

Figure 4.101. Web services export



11.4.4.3. Web Service Import

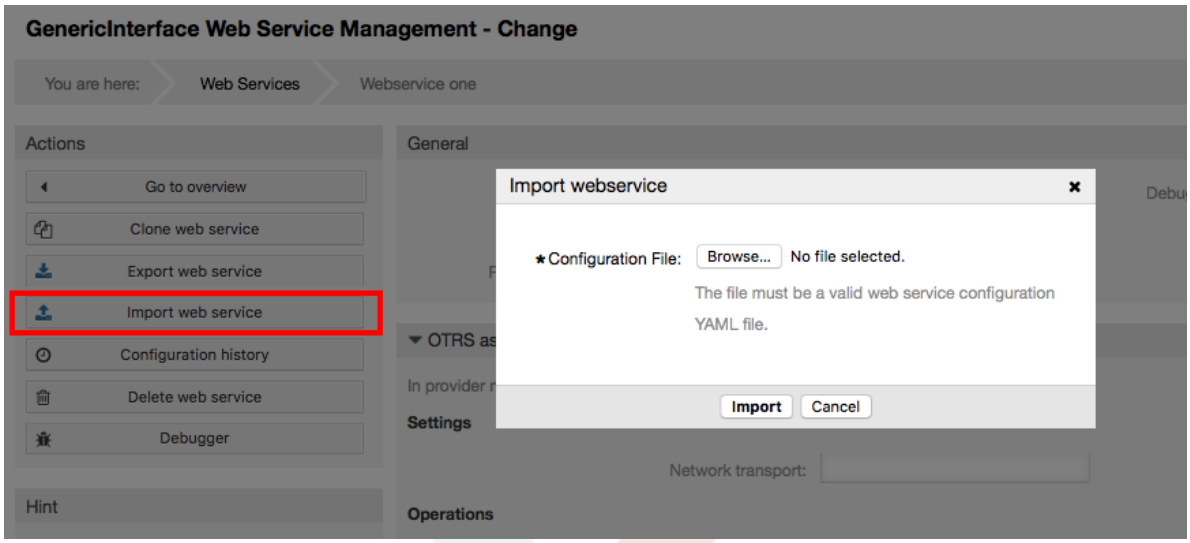
A valid web service configuration YAML file is required to use the import web service feature. Click on the "Import web service" button, browse for the configuration file or provide the complete path in the input box.

Click "Import" button to create a new web service from a file or "Cancel" to close the dialog.

Note

The web service name will be taken from the configuration file name (e.g. if the file name is MyWebservice.yml the resulting web service will be named MyWebservice). If a web service is registered in the system with the same name as the web service that you want to import, the system will lead you to the web service change screen to let you change the name of the imported web service.

Figure 4.102. Web services import



11.4.4.4. Web Service History

Every change to the web service configuration creates a new entry in the web service history (as a journal). The web service history screen displays a list of all configuration versions for a web service. Each row (version) in the "Configuration History List" represents a single revision in the web service history.

Click on one of the rows to show the whole configuration as it was on that particular date / time. The configuration will be shown in the "History details" section of this screen. Here you are also able to export the selected web service configuration version or to restore that version into the current web service configuration.

The "Export web service configuration" behaves exactly as the "Export web service" feature in the web service change screen. For more information refer to that section.

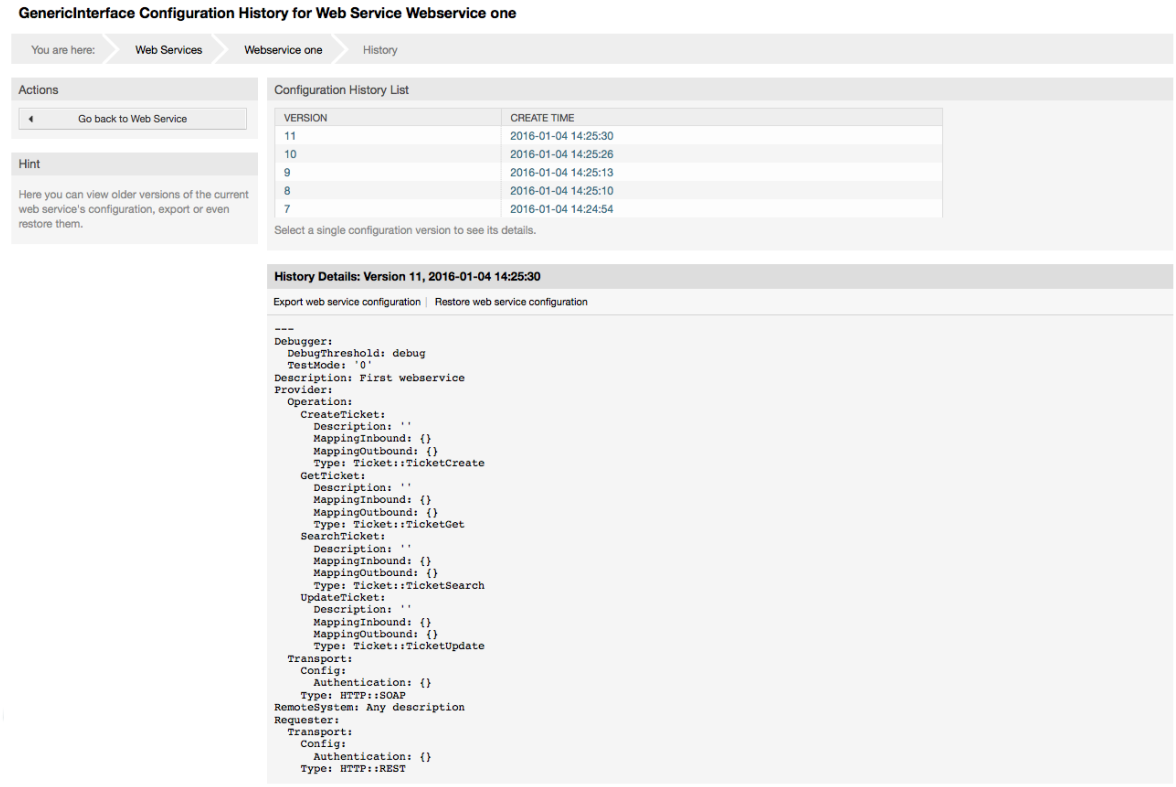
If changes to the current web service configuration do not work as expected and it is not easy to revert the changes manually, you can click on the "Revert web service configuration" button. This will open a dialog to ask you if you are sure to revert the web service configuration. Click "Revert web service configuration" in this dialog to replace the current configuration with the selected version, or click "Cancel" to close the dialog.

Warning

Remember that any passwords stored in the web service configuration will be exported in plain text format.

Please be careful when you restore a configuration because this process is irreversible.

Figure 4.103. Web service history



11.4.4.5. Web Service Delete

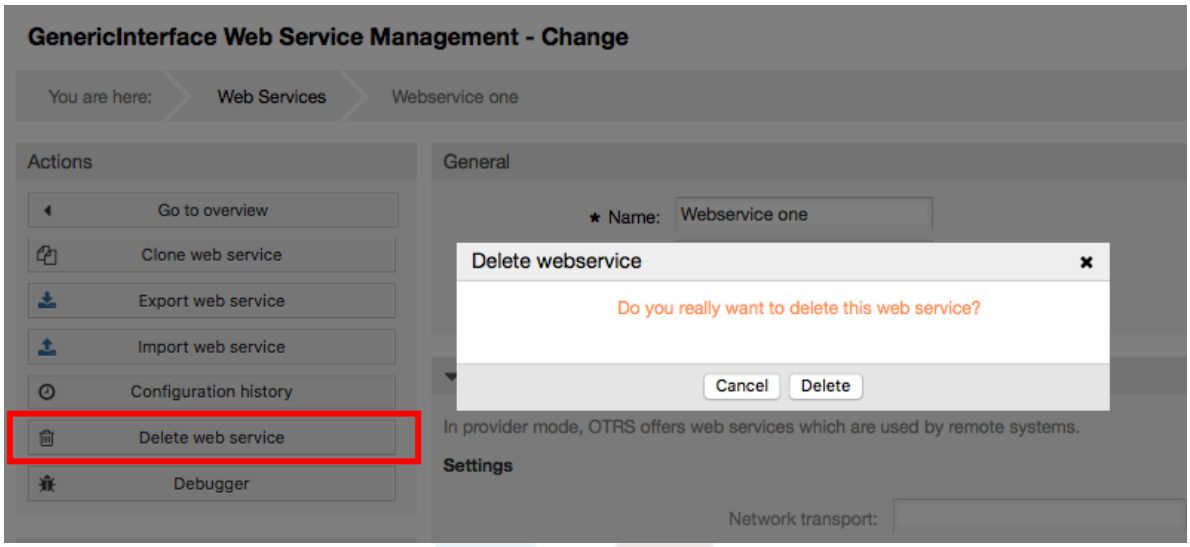
Sometimes it is necessary to delete a web service completely. To do this you can press on the "Delete web service" button and a new dialog will appear asking for confirmation.

Click on "Delete" to confirm the removal of the web service or on "Cancel" to close the dialog.

Warning

Deleting a web service can't be undone, please be careful when deleting a web service.

Figure 4.104. Web service delete



11.4.4.6. Web Service Debugger

The Debugger stores the log of a web service. In the debugger screen you can track all the web service communications for either provider or requester types.

When this screen is shown the request list starts to load. After the list is fully filled you can choose one of the rows (that means a communication sequence) to check its details. This details will appear in a box below.

You can narrow the communication list using the filter on the right part of the screen. You can filter by:

- Communication type (provider or requester)
- Date: before and / or after a particular date
- The remote IP Address
- A combination of all

After filter settings are set, push the "Refresh" button and a new list will be displayed meeting your search criteria.

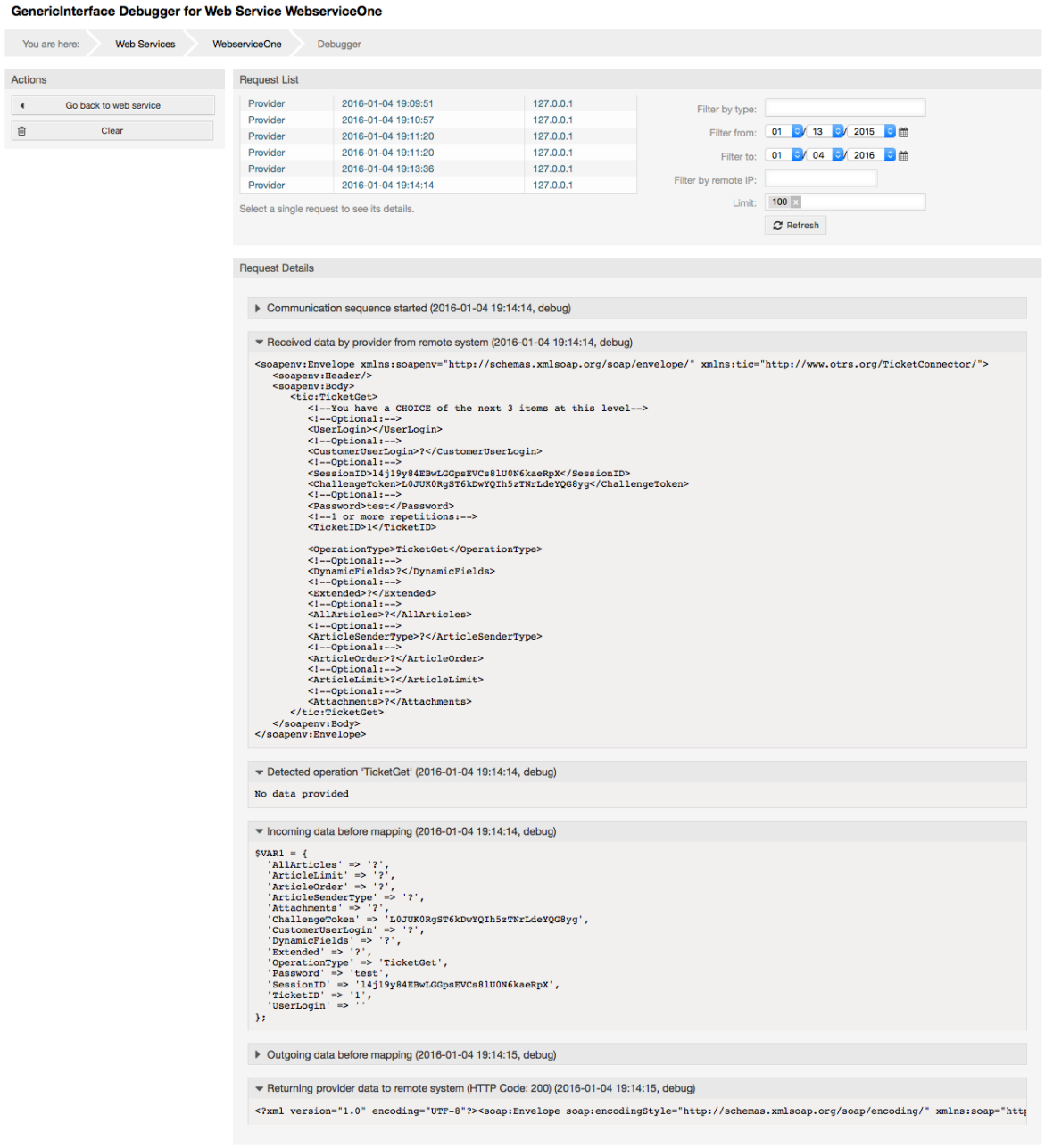
Note

Depending on the search criteria for the filters the new list could return no results.

On the left part of the screen under the action column you can select "Go back to the web service" or clear the debugger log by pushing the "Clear" button. This will open a dialog that ask you to confirm erasing of the log. Click "Clear" in the dialog button to perform the action or click on "Cancel" to close this dialog.

In the "Request details" section you can see all the details for the selected communication. Here you can track the complete flow and check for possible errors or confirm success responses.

Figure 4.105. Web service debugger



11.4.4.7. Web Service Configuration Change

Returning to the web service change screen, now we are going to review the right side of it. Here we have the possibility to modify all the general data for a web service such as name, description, debug threshold, etc. Also there are two more sections below that allows us to modify specific parameters for communication types "OTRS as Provider" and "OTRS as Requester".

The web service configuration needs to be saved on each level. This means that if a setting is changed, links to other, deeper parts of the configuration will be disabled forcing you to save the current configuration level. After saving the disabled links will be re-enabled again allowing you to continue with the configuration.

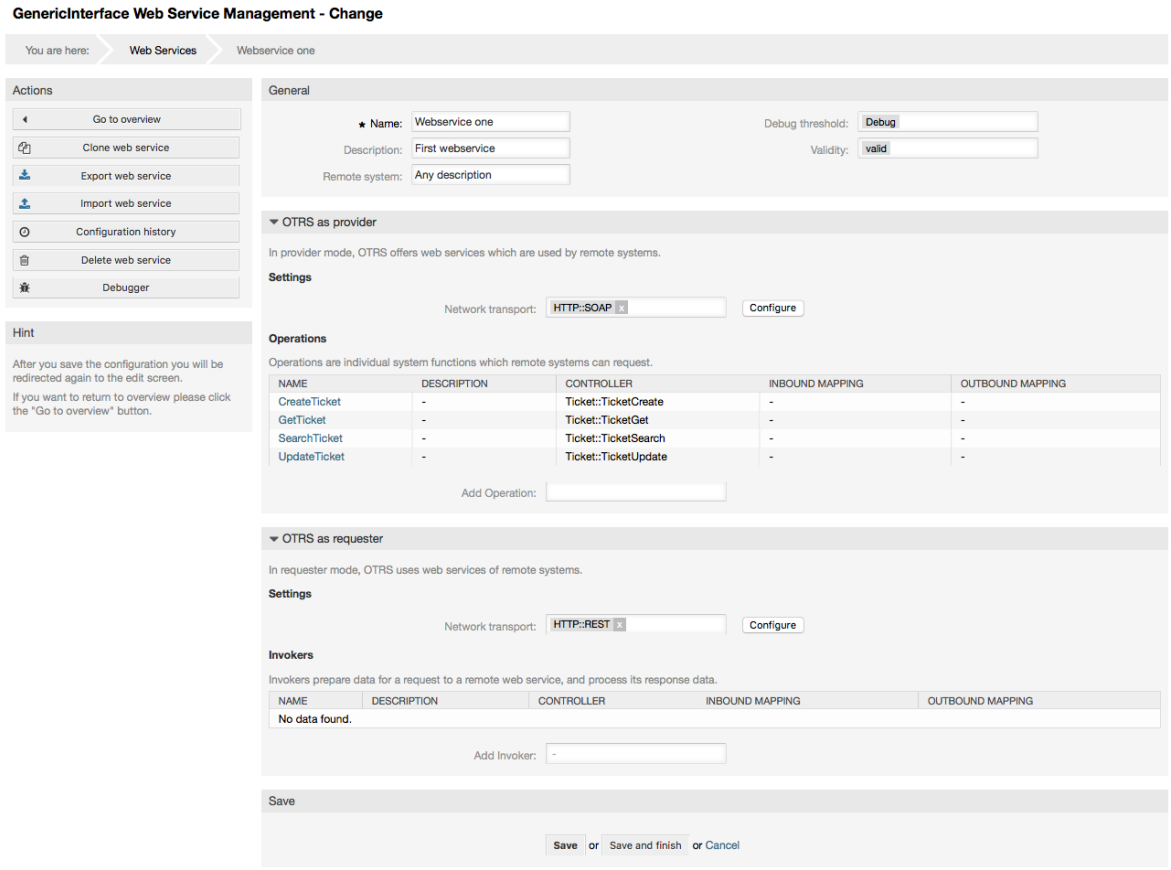
On the "OTRS as provider" section it is possible to set or configure the network transport protocol. Only network transport back-ends that are registered are shown on the list. To

configure the network transport click on the "Configure" button. It is also possible to add new operations in this box. To do this select one of the available operations from the "Add Operation" list. This will lead you to the operation configuration screen. After saving the new operation it will be listed in the table above.

"OTRS as requester" is very similar to the previous one, but instead of "operations" you can add invokers here.

Click the "Save" button to save and continue configuring the web service, "Save and finish" to save and return to the web service overview screen, or "Cancel" to discard current configuration level changes and return to web service overview screen.

Figure 4.106. Web services change



Note

Like the other Generic Interface configuration screens such as Network Transport, Operation, Invoker and Mapping, the initial configuration (add) screen will only present two options: "Save" and "Cancel". If the configuration is re-visited, a new option "Save and Finish" will appear. The behavior of this feature is defined below.

"Save" will store the current configuration level in the database and it will return to the previous screen to review your changes or to configure deeper settings.

"Save and Finish" will store the current configuration level in the database and it will return to the previous screen in the configuration hierarchy (to the immediate upper configuration level).

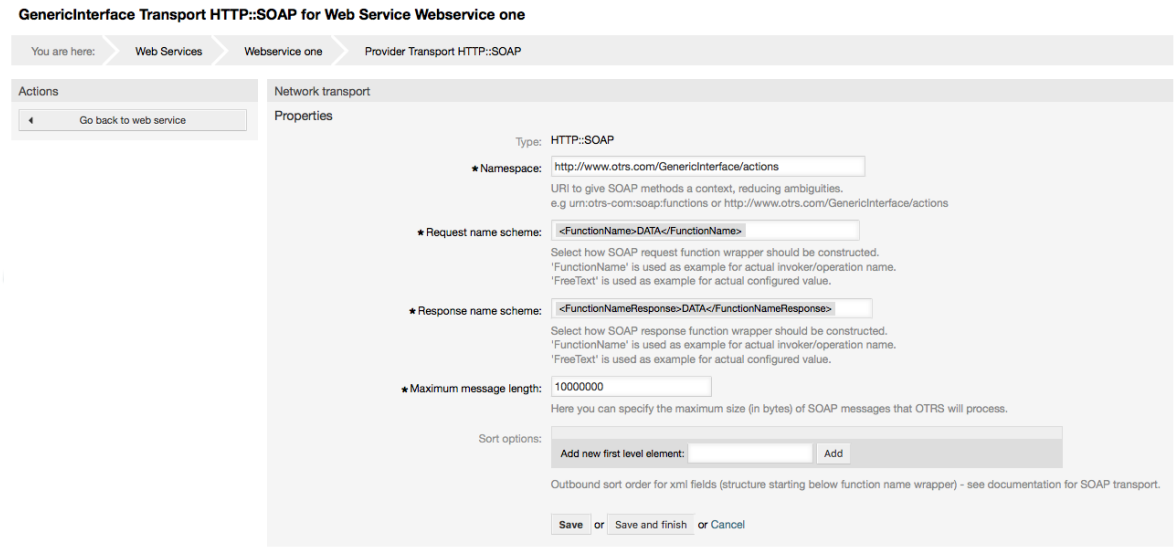
"Cancel" will discard any configuration change to the current configuration level and will return to the previous screen in the configuration hierarchy.

11.4.4.7.1. Web Service Provider Network Transport

In future the list of available network transports will be increased. Currently only "HTTP::SOAP" and "HTTP::REST" transports are available. Each transport has different configuration options to setup and they might use different frontend modules to configure them.

It is quite simple to configure the "HTTP::SOAP" protocol as provider. There are only two settings: "Namespace" and "Maximum message length". These fields are required. The first one is a URI to give SOAP methods a context, reducing ambiguities, and the second one is a field where you can specify the maximum size (in bytes) for SOAP messages that OTRS will process.

Figure 4.107. Web service provider network transport (HTTP::SOAP)



For "HTTP::REST" the configuration might be a bit more complicated, as it grows dynamically for each configured operation by adding: "Route mapping for Operation '<OperationName>':" and "Valid request methods for Operation '<OperationName>':" settings to the default transport settings "Maximum message length:" and "Send Keep-Alive:"

- Route mapping for Operation '<OperationName>':

  In this setting a resource path is set. This path must be defined according to the needs of the web service considering that the path in conjunction with the HTTP request method determines the Generic Interface operation to be executed.

  Path can contain variables in the form of ':<VariableName>' each path string that fits on the position of the variable name will be added to the request payload using the variable name defined in this setting.

  Examples:

  Route mapping: /Resource

  - Valid requests:

    http://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource

    http://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource?Param1=One

- Invalid requests:

  http://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource/

  http://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource/OtherResource

  http://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource/OtherResource?Param1=One

Route mapping: /Resource/:ID

- Valid requests:

  http://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource/1

  http://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource/1?Param1=One

  In both cases ID = 1 will be sent to the operation as part of the payload. In the second case also Param1 = One will be added, depending on the HTTP request method other parameters will be added if they come as a JSON string in the request header.

- Invalid requests:

  http://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource

  http://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource?Param1=One

Route mapping: /Resource/OtherResource/:ID/:Color

- Valid requests:

  http://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource/OtherResource/1/Red

  http://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource/OtherReosurce/123/Blue?Param1=One

  In the first example ID = 1 and Color = Red, while in the second ID = 123 and Color = Blue.

- Invalid requests:

  http://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource/1

  http://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource/OtherResource/1

  http://localhost/otrs/nph-genericinterface.pl/Webservice/Test/Resource/OtherResource/1?Param1=One

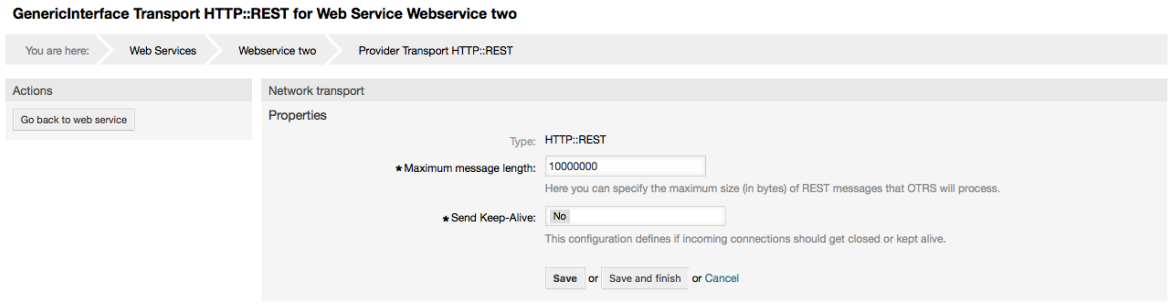
  In the first example the part of the path '/OtherResource' is missing as well as the :Color variable, on the second example just :Color variable is missing.

- Valid request methods for Operation '<OperationName>':

  The HTTP request methods to determine the operation to use together with the route mapping, possible options: CONNECT, DELETE, GET, HEAD, OPTIONS, PATCH, POST, PUT and TRACE.

Totally different operations can share exactly the same mapping path, but the request method must be unique for each operation, in order to determine correctly the operation to use on each request.

Figure 4.108. Web service provider network transport (HTTP::REST)



11.4.4.7.2. Web Service Operation

The actions that can be performed when you are using OTRS as a provider are called "Operations". Each operation belongs to a controller. Controllers are collections of operations or invokers, normally operations from the same controller need similar settings and share the same configuration dialog. But each operation can have independent configuration dialogs if needed.

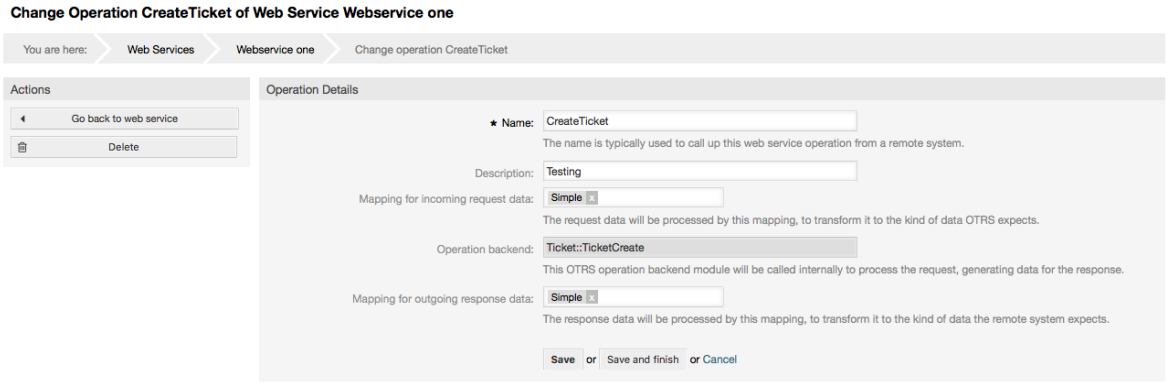
Name, Description, Backend, and Mappings are fields that normally appear on every operation, other special fields can appear in non default configuration dialogs to fulfill specific needs of the operation.

Normally there are two mapping configuration sections on each operation, one for the incoming data and another one for the outgoing data. You can choose different mapping types (backends) for each mapping direction, since their configuration is independent from each other and also independent from the operation backend. The normal and most common practice is that the operation uses the same mapping type in both cases (with inverted configuration). The complete mapping configuration is done in a separate screen which depends on the mapping type.

The operation backend is pre-populated and is not editable. You will see this parameter when you choose the operation on the web service edit screen. The field is only informative.

In the left part of the screen on the action column you have the options: "Go back to web service" (discarding all changes since the last save) and "Delete". If you click on the last one, a dialog will open and ask you if you like to remove the operation. Click on "Delete" button to confirm the removal of the operation and its configuration or "Cancel" to close the delete dialog.

Figure 4.109. Web service operation



11.4.4.7.3. Web Service Requester Network Transport

The network transport configuration for the requester is similar to the configuration for the provider. For the Requester "HTTP::SOAP" network transport there are more fields to be set.

Apart from the "Endpoint" (URI of the Remote System web service interface to accept requests) and "Namespace" which are required fields, you can also specify:

- Encoding (such as utf-8, latin1, iso-8859-1, cp1250, etc) for the SOAP message.
- SOAPAction Header: you can use this to send an empty or filled SOAPAction header. Set to "No" and the SOAPAction header on the SOAP message will be an empty string, or set to "Yes" to send the SOAP action in Namespace#Action format and define the separator (typically "/" for .Net web services and "#" for the REST).
- Authentication: to set the authentication mechanism, set to "-" to not use any authentication or select one from the list and the detail fields will appear.

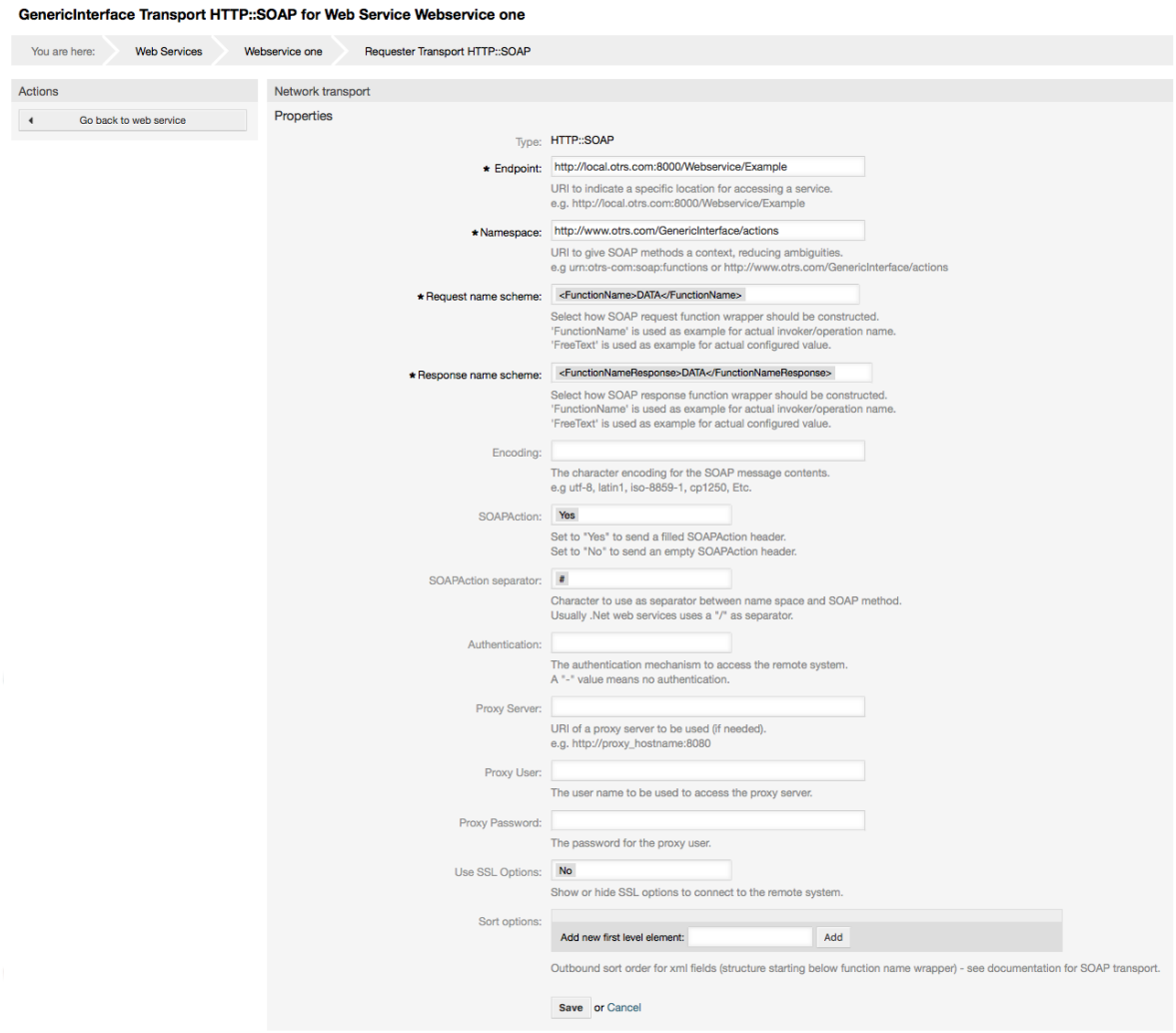
Note

Currently only the "BasicAuth" (HTTP) authentication mechanism is implemented. You can decide whether or not to use it depending on the Remote System configuration. If used, you must provide the User Name and the Password to access the remote system.

Warning

If you supply a password for authentication and after you export the web service to a YAML file this password will be revealed and will be written into a plain text string inside the YAML file. Be aware of it and take precautions if needed.

Figure 4.110. Web service requester network transport (HTTP::SOAP)



In the case of HTTP::REST, this configuration also grows dynamically depending on the configured invokers by adding "Controller mapping for Invoker '<InvokerName>':" and "Valid request command for Invoker '<InvokerName>':" for each invoke. Authentication and SSL options are similar to the ones in HTTP::SOAP

- Host

  The host name or IP Address and port of the remote system, if no port is specified, port 80 is used by default.

- Controller mapping for Invoker '<InvokerName>':

  In this setting a resource path is set. This path must be defined according to the needs of the remote web service and following its definition.

  Path can contain variables in the form of ':<VariableName>' for each variable name that matches the current data (to be sent), will be replaced by the corresponding data value. This matched variable names and values will be removed from the current data. Depending on the HTTP request command the remaining data could be sent as a JSON string in the request body or as query parameters within the URI.

  Examples:

For data: Var1 = One, Var2 = Two, Var3 = Three and Var4 = Four.

Controller mapping: /Resource

- After Replacements:

  /Resource

- Remaining Data:

  Var1 = One, Var2 = Two, Var3 = Three and Var4 = Four

Controller mapping: /Resource/:Var1

- After Replacements:

  /Resource/One

- Remaining Data:

  Var2 = Two, Var3 = Three and Var4 = Four

Controller mapping: /Resource/:Var1?Param1=:Var2&Var3=:Var3

- After Replacements:

  /Resource/One?Param1=Two&Var3=Three

- Remaining Data:

  Var4 = Four

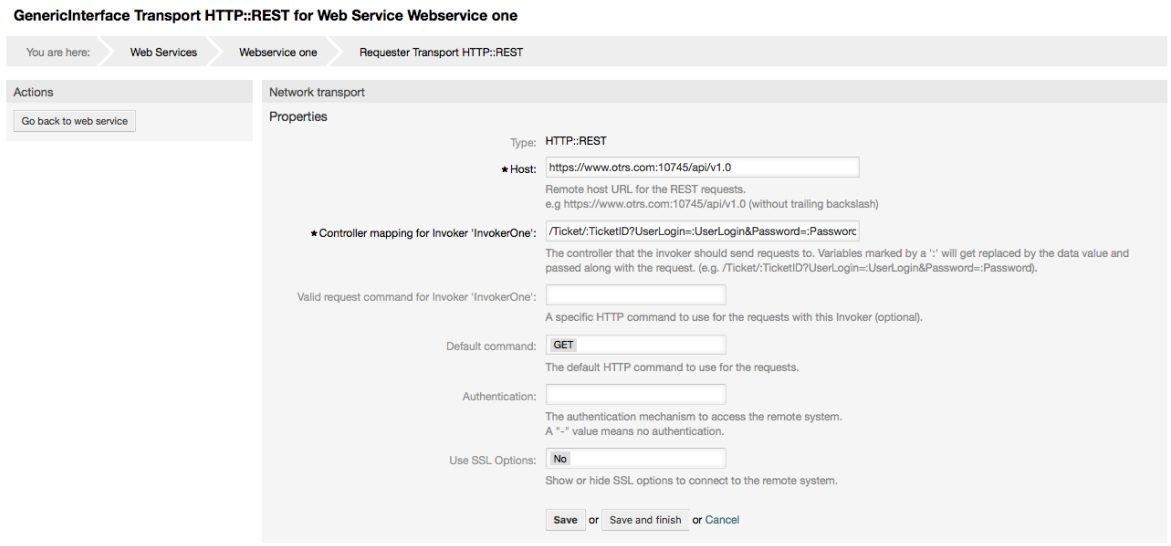
- Valid request command for Invoker '<InvokerName>':

  This determine the HTTP request method to use, possible options: CONNECT, DELETE, GET, HEAD, OPTIONS, PATCH, POST, PUT and TRACE. If no command is selected, Default command is used.

- Default command

  Used as a fall-back for all Invokers without a defined request command.

Figure 4.111. Web service provider network transport (HTTP::REST)



11.4.4.7.4. Web Service Invoker

The actions that can be performed when you are using OTRS as a requester are called "Invokers". Each invoker belongs to a controller (controllers are collections of operations or invokers). Usually invokers from the same controller need similar settings and share the same configuration dialogs. Each invoker can have independent configuration dialogs if needed.

Name, Description, Backend, and Mappings are fields that normally appear on every invoker. Additionally the list of event triggers and other special fields can appear on non default configuration dialogs to fulfill special needs of the invoker.

Normally there are two mapping configuration sections for each invoker, one for the incoming data and another one for the outgoing data. You can choose different mapping types (backends) for each mapping direction, since their configuration is independent from each other and also independent from the invoker backend. The normal and most common practice is that the invoker uses the same mapping type in both cases, with inverted configuration. The complete mapping configuration is done in a separate screen, which depends on the mapping type.

The invoker backend is pre-populated and can not be edited. You will see this parameter when you choose the invoker on the web service edit screen. The field is only informative.

Event triggers are events within OTRS such as "TicketCreate", "ArticleSend", etc. These can act as triggers to execute the invoker. Each invoker needs to have at least one event trigger registered, or the invoker will be useless, because it will never be called. The asynchronous property of the event triggers define if the OTRS process will handle the invoker or if it will be delegated to the OTRS Daemon.

Note

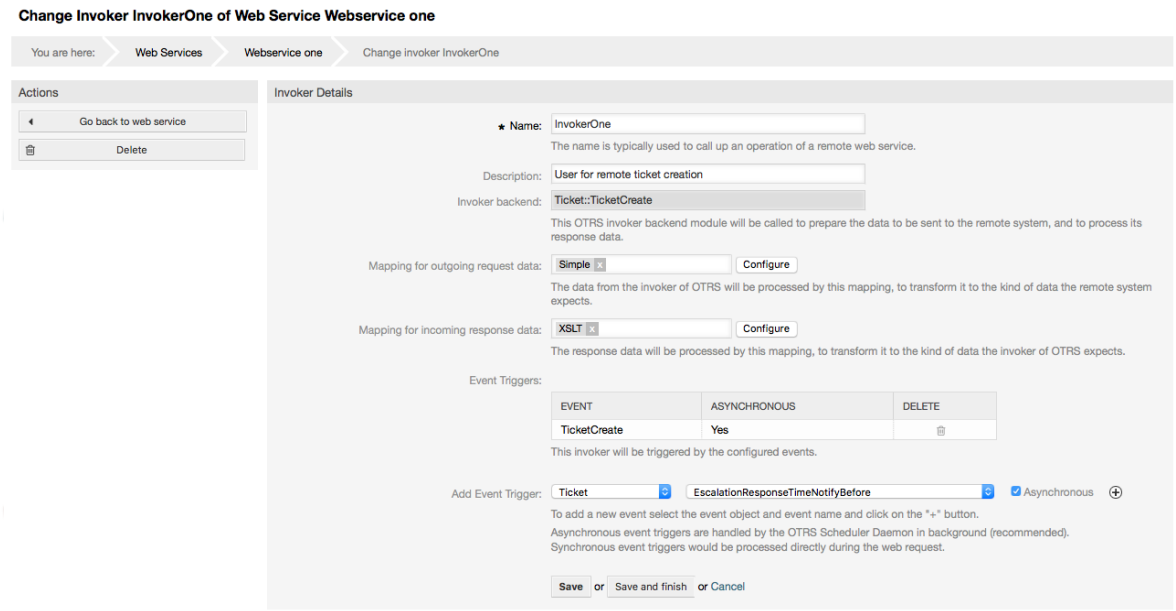
The OTRS Daemon is a separate set of process that executes tasks in the background. Using this the OTRS process itself will not be affected if the Remote System takes a long time to respond, if it is not available or if there are network problems. If you don't use the OTRS Daemons using web services can make OTRS slow or non-responsive. Therefore it is highly recommend to use asynchronous event triggers as often as possible.

To add an Event trigger, first select the event family from the first list, then the event name from the second list, then set the asynchronous property (if unchecked means that the event trigger will not be asynchronous) and finally click on the plus button. A new event trigger will be created and it will be listed on the invoker "Event Triggers" list.

To delete an Event trigger, simply locate the event trigger to be deleted in the "Event Triggers" list and click on the trash icon at the end of the row. This will open a dialog that asks you if you are sure to delete the event trigger. Click "Delete" to remove the event trigger from the list, or "Cancel" to close the dialog.

In the left part of the screen on the action column you have the options: "Go back to web service" (discarding all changes since the last save) and "Delete". If you click on the last one, a dialog will emerge and ask you if you like to remove the invoker. Click on the "Delete" button to confirm the removal of the invoker and its configuration or "Cancel" to close the delete dialog.

Figure 4.112. Web service invoker



11.4.4.7.5. Web Service Mapping

There are cases where you need to transform the data from one format to another (map or change data structure), because normally a web service is used to interact with a Remote System, that is highly probable that is not another OTRS system and / or could not understand the OTRS data structures and values. In these cases some or all values have to be changed, and sometimes even the names of the values (keys) or even the complete structure, in order to match with the expected data on the other end. To accomplish this task the Generic Interface Mapping Layer exists.

Each Remote System has it own data structures and it is possible to create new mapping modules for each case (e.g. there is a customized mapping module for SAP Solution Manager shipped with OTRS), but it is not always necessary. The module Mapping::Simple should cover most of the mapping needs.

Note

When Mapping::Simple does not cover all mapping needs for a web service, a new mapping module should be created. To learn more about how to create new mapping modules please consult the OTRS Development Manual.

This module gives you the opportunity to set default values to map for each key or value for the whole communication data.

At the beginning of the screen you will see a general section where you can set the default rules that will apply for all the unmapped keys and values. There are three options available, these options are listed below:

- Keep (leave unchanged): doesn't touch the keys or values in any way.
- Ignore (drop key/value pair): when this is applied to the key it deletes the key and value, because when a key is deleted then in consequence its associated value is deleted too. When this is applied to the value, only the value is deleted, keeping the key, that now will be associated to an empty value.
- MapTo (use provided key or value as default): all keys and / or values without a defined map rule, will use this as default, when you select this option a new text field will appear to set this default.

Clicking on the "+" button for new key map, will display a new box for a single mapping configuration. You can add as many key mappings as needed. Just click on the "+" button again and a new mapping box will appear below the existing one. From this mapping boxes you can define a map for a single key, with the next options:

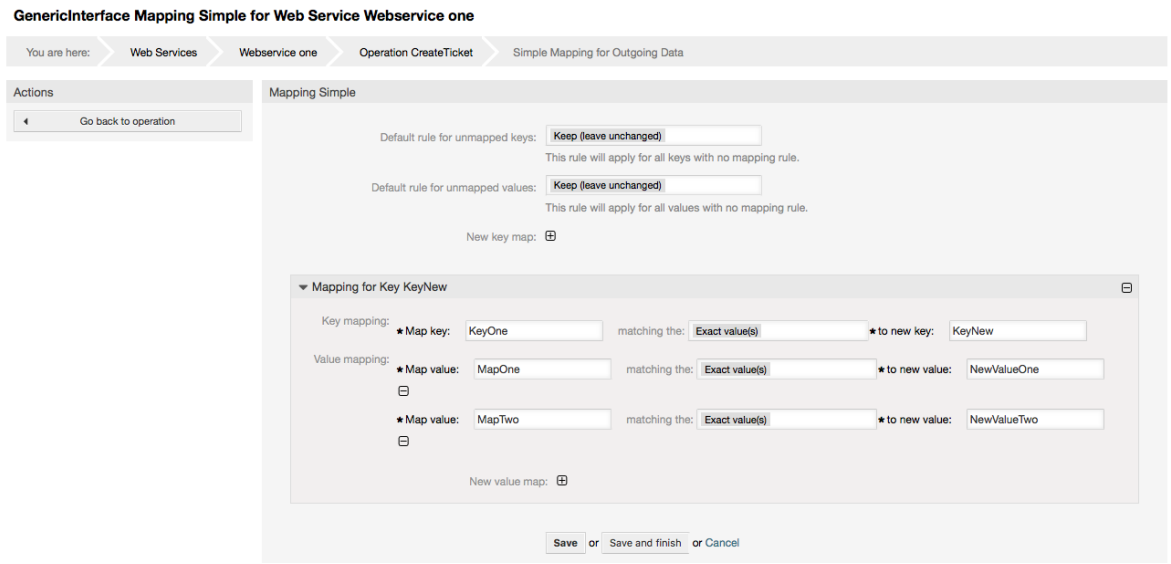
- Exact value(s): the old key string will be changed to a new one if the old key matches exactly.
- Regular expression: the key string will be replaced following a regular expression rule.

Pressing the new value map "+" button will display a new row for a value map. Here it is also possible to define rules for each value to be mapped with the same options as for the key map (Exact value and Regular expression). You can add as many values to map as needed, and if you want to delete one of them, just click on the "-" button for each mapping value row.

Deleting the complete key mapping section (box) is possible, just push on the "-" button located on the up right corner of each box that you want to delete.

If you need to delete a complete mapping configuration: go back to the corresponding operation or invoker screen, look for the mapping direction that you select before and set its value to "-", and save the configuration to apply changes.

Figure 4.113. Web service mapping



11.5. Web Service Command Line Interface

The `bin/otrs.Console.pl Admin::WebService::*` commands were developed in order to create basic, but fast and powerful tools to work with web service configurations. They give you the ability to perform the following actions:

- Add: to create web services using a YAML file as the configuration source.
- Update: to change an existing web service, the configuration can be changed using a different or modified YAML file.
- Dump: to save the current web service configuration to a file.
- List: to get a complete list of all the web services registered in system.
- Delete: to delete a web service from the system. Be careful when you use it, because this action can't be undone.

Example: Creating a new web service configuration:

```
shell> bin/otrs.Console.pl Admin::WebService::Add --name <webservice_name> --source-path /
path/to/yaml/file
```

11.6. Web Service Configuration

From its design the web services were conceived to be portable from one OTRS system to another, e.g. from a test or development environment to a production system. Therefore it was needed to have an easy way to extract the web service configuration from the database, and import it to another. To accomplish this task the Generic Interface uses YAML files as the web services configuration basis.

Why YAML? YAML is a markup language designed to be human friendly to read and write (it is easier to understand than JSON), it does not have some of the limitations of XML like numeric tags, it is open, standardized, and is complete enough to store the whole web service configuration.

Note

To learn more about YAML please visit http://www.yaml.org/.

The following is a web service configuration file example in YAML format:

```
---
Debugger:
  DebugThreshold: debug
Description: This an example of a web service configuration
Provider:
  Operation:
    CloseIncident:
      Description: This is a test operation
      MappingInbound: {}
      MappingOutbound: {}
      RemoteSystemGuid: ''
      Type: Test::Test
    Test:
      Description: This is a test operation
      MappingInbound:
        Config:
          KeyMapDefault:
            MapTo: ''
            MapType: Keep
          KeyMapExact:
```

```
          Prio: Priority
        ValueMap:
          Priority:
            ValueMapExact:
              Critical: 5 Very High
              Information: 1 Very Low
              Warning: 3 Normal
        ValueMapDefault:
          MapTo: 3 Normal
          MapType: MapTo
      Type: Simple
    MappingOutbound:
      Config:
        KeyMapDefault:
          MapTo: ''
          MapType: Ignore
        KeyMapExact:
          Priority: Prio
        ValueMap:
          Prio:
            ValueMapExact:
              1 Very Low: Information
              3 Normal: Warning
              5 Very High: Critical
        ValueMapDefault:
          MapTo: ''
          MapType: Ignore
      Type: Simple
    Type: Test::Test
  Transport:
    Config:
      MaxLength: 10000000
      NameSpace: http://www.example.com/actions
    Type: HTTP::SOAP
RemoteSystem: remote.system.description.example.com
Requester:
  Invoker:
    Test:
      Description: This is a test invoker
      Events:
        - Asynchronous: 1
          Event: TicketCreate
        - Asynchronous: 0
          Event: ArticleUpdate
      MappingInbound:
        Type: Simple
      MappingOutbound:
        Type: Simple
      Type: Test::Test
  Transport:
    Config:
      Authentication:
        Password: '*******'
        Type: BasicAuth
        User: otrs
      Encoding: utf-8
      Endpoint: http://www.example.com:8080/endpoint
      NameSpace: http://www.example.com/actions
      SOAPAction: Yes
      SOAPActionSeparator: '#'
    Type: HTTP::SOAP
```

11.6.1. Configuration Details

11.6.1.1. General

- Description: a short text that describes the web service.
- RemoteSystem: a short description of the Remote System.

- Debugger: a container for the debugger settings.
- Provider: a container for the provider settings.
- Requester: a container for the requester settings.

11.6.1.2. Debugger

- DebugThreshold: the debugger level.

Possible Values

- debug: all logs are stored in the database.
- info: info, notice and error level logs are stored in the database.
- notice: notice and error level logs are stored in the database.
- error: only error level logs are stored in the database.

11.6.1.3. Provider

- Operation: a container for each operation settings.
- Transport: a container for provider network transport settings.

11.6.1.3.1. Operation

- <OperationName>: Unique name for the operation, container for its own operation settings (cardinality 0..n, but not duplicate).

11.6.1.3.1.1. <OperationName>

This section is based on operations from type "Test::Test" other operations might contain more or different settings.

- Description: a short text that describes the operation.
- MappingInbound: a container for the mapping settings for the incoming request data.
- MappingOutbound: a container for the mapping settings for the outgoing response data.
- Type: the operation backend, in Controller::Operation format.

11.6.1.3.1.1.1. MappingInbound

This section is based on mappings from type "Simple". Other mappings might contain more or different settings.

- Config: a container for this mapping settings.
- Type: the mapping backend.

11.6.1.3.1.1.1.1. Config

- KeyMapDefault: a container for all non mapped keys settings.
- ValueMapDefault: a container for all non mapped values settings.
- KeyMapExact: a container for all exact key mappings (cardinality 0 .. 1).
- KeyMapRegEx: a container for all regular expression key mappings (cardinality 0 .. 1).
- ValueMap: a container for all value mappings (cardinality 0 .. 1).

11.6.1.3.1.1.1.1.1. KeyMapDefault

- MapTo: the new value to be used (only applicable if MapType is set to MapTo).
- MapType: the rule for the mapping.

Possible Values

- Keep: leave unchanged.
- Ignore: drop.
- MapTo: change to the MapTo value.

11.6.1.3.1.1.1.1.2. ValueMapDefault

Similar to KeyMapDefault.

11.6.1.3.1.1.1.1.3. KeyMapExact

- <oldkey>: <newkey> (cardinality 0 .. n but not duplicate).

11.6.1.3.1.1.1.1.4. KeyMapRegEx

- <oldkey(RegEx)>: <newkey> (cardinality 0 .. n but no duplicates).

11.6.1.3.1.1.1.1.5. ValueMap

- <newkey>: a container for value mappings for this new key (cardinality depends on the new keys from KeyMapExact and KeyMapRegEx).

11.6.1.3.1.1.1.1.5.1. <newkey>

- ValueMapExact: a container for all exact value mappings (cardinality 0 .. 1).
- ValueMapRegEx: a container for all regular expression value mappings (cardinality 0 .. 1).

11.6.1.3.1.1.1.1.5.1.1. ValueMapExact

- <oldvalue>: <newvalue> (cardinality 0 .. n but not duplicate).

11.6.1.3.1.1.1.1.5.1.2. ValueMapRegEx

- <oldvalue(RegEx)>: <newvalue> (cardinality 0 .. n but not duplicate).

11.6.1.3.1.1.2. MappingOutbound

Same as MappingInbound.

11.6.1.3.1.1.3. Transport

This section is based on the provider network transport HTTP::SOAP, other transports might contain more or different settings.

- Config: a container for the specific network transport configuration settings.
- Type: the provider network transport backend.

11.6.1.3.1.1.3.1. Config

- MaxLength: the maximum length in bytes to be read in a SOAP message by OTRS.
- NameSpace: an URI that gives a context to all operations that belongs to this web service.

11.6.1.4. Requester

- Invoker: a container for each invokers' settings.
- Transport: a container for requester network transport settings.

11.6.1.4.1. Invoker

- <InvokerName>: Unique name for the invoker, container for its own invoker settings (cardinality 0..n, but not duplicate).

11.6.1.4.1.1. <InvokerName>

This section is based on invokers from type "Test::Test" other invokers might contain more or different settings.

- Description: a short text that describes the invoker.
- Events: a container for a unnamed list of event trigger settings.
- MappingInbound: a container for the mapping settings for the incoming response data.
- MappingOutbound: a container for the mapping settings for the outgoing request data.
- Type: the invoker backend, in Controller::Invoker format.

11.6.1.4.1.1.1. Events

- *List Element*: (cardinality 0 .. n).
  - Asynchronous: to set if the invoker execution will be delegated to the OTRS Daemon.

    Possible Values

    - 0: not handled by the OTRS Daemon.
    - 1: handled by the OTRS Daemon.
  - Event: the name of the event trigger.

    Possible Values (for ticket events)

    - TicketCreate
    - TicketDelete
    - TicketTitleUpdate
    - TicketUnlockTimeoutUpdate
    - TicketQueueUpdate
    - TicketTypeUpdate
    - TicketServiceUpdate
    - TicketSLAUpdate
    - TicketCustomerUpdate
    - TicketFreeTextUpdate
    - TicketFreeTimeUpdate

- TicketPendingTimeUpdate
- TicketLockUpdate
- TicketArchiveFlagUpdate
- TicketStateUpdate
- TicketOwnerUpdate
- TicketResponsibleUpdate
- TicketPriorityUpdate
- HistoryAdd
- HistoryDelete
- TicketAccountTime
- TicketMerge
- TicketSubscribe
- TicketUnsubscribe
- TicketFlagSet
- TicketFlagDelete
- TicketSlaveLinkAdd
- TicketSlaveLinkDelete
- TicketMasterLinkDelete

Possible Values (for article events)

- ArticleCreate
- ArticleFreeTextUpdate
- ArticleUpdate
- ArticleSend
- ArticleBounce
- ArticleAgentNotification
- ArticleCustomerNotification
- ArticleAutoResponse
- ArticleFlagSet
- ArticleFlagDelete
- ArticleAgentNotification
- ArticleCustomerNotification

11.6.1.4.1.1.2. MappingInbound

Same as Operation MappingInbound.

11.6.1.4.1.1.3. MappingOutbound

Same as Operation MappingInbound.

11.6.1.4.1.1.4. Transport

This section is based on the requester network transport HTTP::SOAP, other transports might contain more or different settings.

- Config: a container for the specific network transport configuration settings.
- Type: the requester network transport backend.

11.6.1.4.1.1.4.1. Config

- Authentication: a container for authentication settings.
- Encoding: the SOAP Message request encoding.
- Endpoint: the URI of the Remote Server web service to accept OTRS requests.
- NameSpace: an URI that gives a context to all invokers that belongs to this web service.
- SOAPAction: to send an empty or filled SOAPAction header in the SOAP Message (in "<NameSpace> <Separator> <Action>" format).

  Possible Values

  - Yes: to send a filled SOAPAction header.
  - No: to send an empty SOAPAction header.
- SOAPActionSeparator: to set the <Separator> of a filled SOAPAction header.

  Possible Values

  - '/': used for .net web services.
  - '#': used for all the rest web services.

11.6.1.4.1.1.4.1.1. Authentication

- User: the privileged user name that has access to the remote web service.
- Password: the password for privileged user in plain text.
- Type: the type of authentication.

11.7. Connectors

A Connector is in essence a set of actions that are either called Operations if OTRS acts as a web service provider or Invokers if OTRS acts as a web service requester. But it can also include special Mappings or Transports.

One Connector can either have only Operations, Only Invokers or both. A connector can even use parts of other connectors like the Mappings or Transports if they are not to specific for the Connector that is trying to implement them.

In other words a Connector is not limited to just the Controller layer but it can be extended to Data Mapping or Network Transport layers if needed.

Due to the modular design of the Generic Interface a Connector can be seen as a plug-in; this means that by adding Connectors the capabilities of the generic interface can be extended using: OTRS Feature add ons, OTRS Custom modules, 3rd Party modules, and so on.

11.7.1. Bundled Connectors

Included with this version of OTRS the following connectors are ready to be used:

- Session
- Ticket

11.7.1.1. Session Connector

This connector is capable to create a valid SessionID that can be used in any other operation.

Provides:

- Operations:
  - SessionCreate

11.7.1.1.1. Operations

11.7.1.1.1.1. SessionCreate

Creates a new valid SessionID to be used in other operations from other connectors like TicketCreate.

> **Note**
>
> To use the SessionID in other operations from other connectors it is necessary that the operation implements authentication by SessionID. All the rest of the bundled operations are capable of accepting a valid SessionID as an authentication method.

Possible Attributes:

```
<SessionCreate>
   <!--You have a MANDATORY CHOICE of the next 2 items at this level-->
   <!--Optional:-->
   <UserLogin>?</UserLogin>
   <!--Optional:-->
   <CustomerUserLogin>?</CustomerUserLogin>
   <!--Optional:-->
   <Password>?</Password>
</SessionCreate>
```

11.7.1.2. Ticket Connector

This connector supplies the basic functionality to interact with tickets.

Provides:

- Operations:
  - TicketCreate
  - TicketUpdate

- TicketGet
- TicketSearch

11.7.1.2.1. Operations

11.7.1.2.1.1. TicketCreate

Provides an interface to create a ticket in OTRS. A ticket must contain an Article and can contain several attachments, all defined Dynamic Fields can be also set on TicketCreate operation.

Possible Attributes:

```
<TicketCreate>
   <!--You have a MANDATORY CHOICE of the next 3 items at this level-->
   <!--Optional:-->
   <UserLogin>?</UserLogin>
   <!--Optional:-->
   <CustomerUserLogin>?</CustomerUserLogin>
   <!--Optional:-->
   <SessionID>?</SessionID>
   <!--Optional:-->
   <Password>?</Password>
   <Ticket>
      <Title>?</Title>
      <!--You have a MANDATORY CHOICE of the next 2 items at this level-->
      <!--Optional:-->
      <QueueID>?</QueueID>
      <!--Optional:-->
      <Queue>?</Queue>
      <!--You have a CHOICE of the next 2 items at this level-->
      <!--Optional:-->
      <TypeID>?</TypeID>
      <!--Optional:-->
      <Type>?</Type>
      <!--You have a CHOICE of the next 2 items at this level-->
      <!--Optional:-->
      <ServiceID>?</ServiceID>
      <!--Optional:-->
      <Service>?</Service>
      <!--You have a CHOICE of the next 2 items at this level-->
      <!--Optional:-->
      <SLAID>?</SLAID>
      <!--Optional:-->
      <SLA>?</SLA>
      <!--You have a MANDATORY CHOICE of the next 2 items at this level-->
      <!--Optional:-->
      <StateID>?</StateID>
      <!--Optional:-->
      <State>?</State>
      <!--You have a MANDATORY CHOICE of the next 2 items at this level-->
      <!--Optional:-->
      <PriorityID>?</PriorityID>
      <!--Optional:-->
      <Priority>?</Priority>
      <!--You have a CHOICE of the next 2 items at this level-->
      <!--Optional:-->
      <OwnerID>?</OwnerID>
      <!--Optional:-->
      <Owner>?</Owner>
      <!--You have a CHOICE of the next 2 items at this level-->
      <!--Optional:-->
      <ResponsibleID>?</ResponsibleID>
      <!--Optional:-->
      <Responsible>?</Responsible>
      <CustomerUser>?</CustomerUser>
```

```
            <!--Optional:-->
            <CustomerID>?</CustomerID>
            <!--Optional:-->
            <PendingTime>
               <!--You have a CHOICE of the next and the other 5 items at this level-->
               <Diff>?</Diff>
               <Year>?</Year>
               <Month>?</Month>
               <Day>?</Day>
               <Hour>?</Hour>
               <Minute>?</Minute>
            </PendingTime>
         </Ticket>
         <Article>
            <!--You have a CHOICE of the next 2 items at this level-->
            <!--Optional:-->
            <ArticleTypeID>?</ArticleTypeID>
            <!--Optional:-->
            <ArticleType>?</ArticleType>
            <!--You have a CHOICE of the next 2 items at this level-->
            <!--Optional:-->
            <SenderTypeID>?</SenderTypeID>
            <!--Optional:-->
            <SenderType>?</SenderType>
            <!--Optional:-->
            <From>?</From>
            <Subject>?</Subject>
            <Body>?</Body>
            <!--You have a CHOICE of the next 2 items at this level-->
            <!--Optional:-->
            <ContentType>?</ContentType>
            <Charset>?</Charset>
            <MimeType>?</MimeType>
            <!--Optional:-->
            <HistoryType>?</HistoryType>
            <!--Optional:-->
            <HistoryComment>?</HistoryComment>
            <!--Optional:-->
            <AutoResponseType>?</AutoResponseType>
            <!--Optional:-->
            <TimeUnit>?</TimeUnit>
            <!--Optional:-->
            <NoAgentNotify>?</NoAgentNotify>
            <!--Zero or more repetitions:-->
            <ForceNotificationToUserID>?</ForceNotificationToUserID>
            <!--Zero or more repetitions:-->
            <ExcludeNotificationToUserID>?</ExcludeNotificationToUserID>
            <!--Zero or more repetitions:-->
            <ExcludeMuteNotificationToUserID>?</ExcludeMuteNotificationToUserID>
         </Article>
         <!--Zero or more repetitions:-->
         <DynamicField>
            <Name>?</Name>
            <!--1 or more repetitions:-->
            <Value>?</Value>
         </DynamicField>
         <!--Zero or more repetitions:-->
         <Attachment>
            <Content>cid:61886944659</Content>
            <ContentType>?</ContentType>
            <Filename>?</Filename>
         </Attachment>
      </TicketCreate>
```

11.7.1.2.1.2. TicketUpdate

TicketUpdate operation adds the capability to modify attributes from an existing ticket or to add a new article, including attachments and all defined dynamic fields for the ticket and the new article.

Note

It is not necessary to create a new article to modify a ticket attribute.

Possible Attributes:

```
<TicketUpdate>
   <!--You have a MANDATORY CHOICE of the next 3 items at this level-->
   <!--Optional:-->
   <UserLogin>?</UserLogin>
   <!--Optional:-->
   <CustomerUserLogin>?</CustomerUserLogin>
   <!--Optional:-->
   <SessionID>?</SessionID>
   <!--Optional:-->
   <Password>?</Password>
   <!--You have a CHOICE of the next 2 items at this level-->
   <TicketID>?</TicketID>
   <TicketNumber>?</TicketNumber>
   <!--Optional:-->
   <Ticket>
      <!--Optional:-->
      <Title>?</Title>
      <!--You have a CHOICE of the next 2 items at this level-->
      <!--Optional:-->
      <QueueID>?</QueueID>
      <!--Optional:-->
      <Queue>?</Queue>
      <!--You have a CHOICE of the next 2 items at this level-->
      <!--Optional:-->
      <TypeID>?</TypeID>
      <!--Optional:-->
      <Type>?</Type>
      <!--You have a CHOICE of the next 2 items at this level-->
      <!--Optional:-->
      <ServiceID>?</ServiceID>
      <!--Optional:-->
      <Service>?</Service>
      <!--You have a CHOICE of the next 2 items at this level-->
      <!--Optional:-->
      <SLAID>?</SLAID>
      <!--Optional:-->
      <SLA>?</SLA>
      <!--You have a CHOICE of the next 2 items at this level-->
      <!--Optional:-->
      <StateID>?</StateID>
      <!--Optional:-->
      <State>?</State>
      <!--You have a CHOICE of the next 2 items at this level-->
      <!--Optional:-->
      <PriorityID>?</PriorityID>
      <!--Optional:-->
      <Priority>?</Priority>
      <!--You have a CHOICE of the next 2 items at this level-->
      <!--Optional:-->
      <OwnerID>?</OwnerID>
      <!--Optional:-->
      <Owner>?</Owner>
      <!--You have a CHOICE of the next 2 items at this level-->
      <!--Optional:-->
      <ResponsibleID>?</ResponsibleID>
      <!--Optional:-->
      <Responsible>?</Responsible>
      <!--Optional:-->
      <CustomerUser>?</CustomerUser>
      <!--Optional:-->
      <CustomerID>?</CustomerID>
      <!--Optional:-->
      <PendingTime>
```

```
            <!--You have a CHOICE of the next and the other 5 items at this level-->
            <Diff>?</Diff>
            <Year>?</Year>
            <Month>?</Month>
            <Day>?</Day>
            <Hour>?</Hour>
            <Minute>?</Minute>
         </PendingTime>
      </Ticket>
      <!--Optional:-->
      <Article>
         <!--You have a CHOICE of the next 2 items at this level-->
         <!--Optional:-->
         <ArticleTypeID>?</ArticleTypeID>
         <!--Optional:-->
         <ArticleType>?</ArticleType>
         <!--You have a CHOICE of the next 2 items at this level-->
         <!--Optional:-->
         <SenderTypeID>?</SenderTypeID>
         <!--Optional:-->
         <SenderType>?</SenderType>
         <!--Optional:-->
         <From>?</From>
         <Subject>?</Subject>
         <Body>?</Body>
         <!--You have a CHOICE of the next 2 items at this level-->
         <!--Optional:-->
         <ContentType>?</ContentType>
         <Charset>?</Charset>
         <MimeType>?</MimeType>
         <!--Optional:-->
         <HistoryType>?</HistoryType>
         <!--Optional:-->
         <HistoryComment>?</HistoryComment>
         <!--Optional:-->
         <AutoResponseType>?</AutoResponseType>
         <!--Optional:-->
         <TimeUnit>?</TimeUnit>
         <!--Optional:-->
         <NoAgentNotify>?</NoAgentNotify>
         <!--Zero or more repetitions:-->
         <ForceNotificationToUserID>?</ForceNotificationToUserID>
         <!--Zero or more repetitions:-->
         <ExcludeNotificationToUserID>?</ExcludeNotificationToUserID>
         <!--Zero or more repetitions:-->
         <ExcludeMuteNotificationToUserID>?</ExcludeMuteNotificationToUserID>
      </Article>
      <!--Zero or more repetitions:-->
      <DynamicField>
         <Name>?</Name>
         <!--1 or more repetitions:-->
         <Value>?</Value>
      </DynamicField>
      <!--Zero or more repetitions:-->
      <Attachment>
         <Content>cid:166861569966</Content>
         <ContentType>?</ContentType>
         <Filename>?</Filename>
      </Attachment>
   </TicketUpdate>
```

11.7.1.2.1.3. TicketGet

This operation is used to get all the attributes of a ticket including the dynamic fields, all articles and all of the attachments that belong to the ticket.

Possible Attributes:

```
```

```
<TicketGet>
   <!--You have a MANDATORY CHOICE of the next 3 items at this level-->
   <!--Optional:-->
   <UserLogin>?</UserLogin>
   <!--Optional:-->
   <CustomerUserLogin>?</CustomerUserLogin>
   <!--Optional:-->
   <SessionID>?</SessionID>
   <!--Optional:-->
   <Password>?</Password>
   <!--1 or more repetitions:-->
   <TicketID>?</TicketID>
   <!--Optional:-->
   <DynamicFields>?</DynamicFields>
   <!--Optional:-->
   <Extended>?</Extended>
   <!--Optional:-->
   <AllArticles>?</AllArticles>
   <!--Optional:-->
   <ArticleSenderType>?</ArticleSenderType>
   <!--Optional:-->
   <ArticleOrder>?</ArticleOrder>
   <!--Optional:-->
   <ArticleLimit>?</ArticleLimit>
   <!--Optional:-->
   <Attachments>?</Attachments>
   <!--Optional:-->
   <HTMLBodyAsAttachment>?</HTMLBodyAsAttachment>
</TicketGet>
```

11.7.1.2.1.4. TicketSearch

TicketSearch operation returns a list of Ticket IDs that matches a predefined criteria.

Possible Attributes:

```
<TicketSearch>
   <!--You have a MANDATORY CHOICE of the next 3 items at this level-->
   <!--Optional:-->
   <UserLogin>?</UserLogin>
   <!--Optional:-->
   <CustomerUserLogin>?</CustomerUserLogin>
   <!--Optional:-->
   <SessionID>?</SessionID>
   <!--Optional:-->
   <Password>?</Password>
   <!--Optional:-->
   <Limit>?</Limit>
   <!--Zero or more repetitions:-->
   <TicketNumber>?</TicketNumber>
   <!--Zero or more repetitions:-->
   <Title>?</Title>
   <!--Zero or more repetitions:-->
   <Queues>?</Queues>
   <!--Zero or more repetitions:-->
   <QueueIDs>?</QueueIDs>
   <!--Optional:-->
   <UseSubQueues>?</UseSubQueues>
   <!--Zero or more repetitions:-->
   <Types>?</Types>
   <!--Zero or more repetitions:-->
   <TypeIDs>?</TypeIDs>
   <!--Zero or more repetitions:-->
   <States>?</States>
   <!--Zero or more repetitions:-->
   <StateIDs>?</StateIDs>
   <!--Zero or more repetitions:-->
   <StateType>?</StateType>
```

```
        <!--Zero or more repetitions:-->
        <StateTypeIDs>?</StateTypeIDs>
        <!--Zero or more repetitions:-->
        <Priorities>?</Priorities>
        <!--Zero or more repetitions:-->
        <PriorityIDs>?</PriorityIDs>
        <!--Zero or more repetitions:-->
        <Services>?</Services>
        <!--Zero or more repetitions:-->
        <ServiceIDs>?</ServiceIDs>
        <!--Zero or more repetitions:-->
        <SLAs>?</SLAs>
        <!--Zero or more repetitions:-->
        <SLAIDs>?</SLAIDs>
        <!--Zero or more repetitions:-->
        <Locks>?</Locks>
        <!--Zero or more repetitions:-->
        <LockIDs>?</LockIDs>
        <!--Zero or more repetitions:-->
        <OwnerIDs>?</OwnerIDs>
        <!--Zero or more repetitions:-->
        <ResponsibleIDs>?</ResponsibleIDs>
        <!--Zero or more repetitions:-->
        <WatchUserIDs>?</WatchUserIDs>
        <!--Zero or more repetitions:-->
        <CustomerID>?</CustomerID>
        <!--Zero or more repetitions:-->
        <CustomerUserLogin>?</CustomerUserLogin>
        <!--Zero or more repetitions:-->
        <CreatedUserIDs>?</CreatedUserIDs>
        <!--Zero or more repetitions:-->
        <CreatedTypes>?</CreatedTypes>
        <!--Zero or more repetitions:-->
        <CreatedTypeIDs>?</CreatedTypeIDs>
        <!--Zero or more repetitions:-->
        <CreatedPriorities>?</CreatedPriorities>
        <!--Zero or more repetitions:-->
        <CreatedPriorityIDs>?</CreatedPriorityIDs>
        <!--Zero or more repetitions:-->
        <CreatedStates>?</CreatedStates>
        <!--Zero or more repetitions:-->
        <CreatedStateIDs>?</CreatedStateIDs>
        <!--Zero or more repetitions:-->
        <CreatedQueues>?</CreatedQueues>
        <!--Zero or more repetitions:-->
        <CreatedQueueIDs>?</CreatedQueueIDs>
        <!--Zero or more repetitions:-->
        <DynamicFields>
           <!--You have a MANDATORY CHOICE of the next 6 items at this level-->
           <!--Optional:-->
           <Equals>?</Equals>
           <!--Optional:-->
           <Like>?</Like>
           <!--Optional:-->
           <GreaterThan>?</GreaterThan>
           <!--Optional:-->
           <GreaterThanEquals>?</GreaterThanEquals>
           <!--Optional:-->
           <SmallerThan>?</SmallerThan>
           <!--Optional:-->
           <SmallerThanEquals>?</SmallerThanEquals>
        </DynamicFields>
        <!--Optional:-->
        <Ticketflag>
           <!--Optional:-->
           <Seen>?</Seen>
        </Ticketflag>
        <!--Optional:-->
        <From>?</From>
        <!--Optional:-->
        <To>?</To>
        <!--Optional:-->
```

```
        <Cc>?</Cc>
        <!--Optional:-->
        <Subject>?</Subject>
        <!--Optional:-->
        <Body>?</Body>
        <!--Optional:-->
        <FullTextIndex>?</FullTextIndex>
        <!--Optional:-->
        <ContentSearch>?</ContentSearch>
        <!--Optional:-->
        <ConditionInline>?</ConditionInline>
        <!--Optional:-->
        <ArticleCreateTimeOlderMinutes>?</ArticleCreateTimeOlderMinutes>
        <!--Optional:-->
        <ArticleCreateTimeNewerMinutes>?</ArticleCreateTimeNewerMinutes>
        <!--Optional:-->
        <ArticleCreateTimeNewerDate>?</ArticleCreateTimeNewerDate>
        <!--Optional:-->
        <ArticleCreateTimeOlderDate>?</ArticleCreateTimeOlderDate>
        <!--Optional:-->
        <TicketCreateTimeOlderMinutes>?</TicketCreateTimeOlderMinutes>
        <!--Optional:-->
        <ATicketCreateTimeNewerMinutes>?</ATicketCreateTimeNewerMinutes>
        <!--Optional:-->
        <TicketCreateTimeNewerDate>?</TicketCreateTimeNewerDate>
        <!--Optional:-->
        <TicketCreateTimeOlderDate>?</TicketCreateTimeOlderDate>
        <!--Optional:-->
        <TicketLastChangeTimeOlderMinutes>?</TicketLastChangeTimeOlderMinutes>
        <!--Optional:-->
        <TicketLastChangeTimeNewerMinutes>?</TicketLastChangeTimeNewerMinutes>
        <!--Optional:-->
        <TicketLastChangeTimeNewerDate>?</TicketLastChangeTimeNewerDate>
        <!--Optional:-->
        <TicketLastChangeTimeOlderDate>?</TicketLastChangeTimeOlderDate>
        <!--Optional:-->
        <TicketChangeTimeOlderMinutes>?</TicketChangeTimeOlderMinutes>
        <!--Optional:-->
        <TicketChangeTimeNewerMinutes>?</TicketChangeTimeNewerMinutes>
        <!--Optional:-->
        <TicketChangeTimeNewerDate>?</TicketChangeTimeNewerDate>
        <!--Optional:-->
        <TicketChangeTimeOlderDate>?</TicketChangeTimeOlderDate>
        <!--Optional:-->
        <TicketCloseTimeOlderMinutes>?</TicketCloseTimeOlderMinutes>
        <!--Optional:-->
        <TicketCloseTimeNewerMinutes>?</TicketCloseTimeNewerMinutes>
        <!--Optional:-->
        <TicketCloseTimeNewerDate>?</TicketCloseTimeNewerDate>
        <!--Optional:-->
        <TicketCloseTimeOlderDate>?</TicketCloseTimeOlderDate>
        <!--Optional:-->
        <TicketPendingTimeOlderMinutes>?</TicketPendingTimeOlderMinutes>
        <!--Optional:-->
        <TicketPendingTimeNewerMinutes>?</TicketPendingTimeNewerMinutes>
        <!--Optional:-->
        <TicketPendingTimeNewerDate>?</TicketPendingTimeNewerDate>
        <!--Optional:-->
        <TicketPendingTimeOlderDate>?</TicketPendingTimeOlderDate>
        <!--Optional:-->
        <TicketEscalationTimeOlderMinutes>?</TicketEscalationTimeOlderMinutes>
        <!--Optional:-->
        <TTicketEscalationTimeNewerMinutes>?</TTicketEscalationTimeNewerMinutes>
        <!--Optional:-->
        <TicketEscalationTimeNewerDate>?</TicketEscalationTimeNewerDate>
        <!--Optional:-->
        <TicketEscalationTimeOlderDate>?</TicketEscalationTimeOlderDate>
        <!--Optional:-->
        <ArchiveFlags>?</ArchiveFlags>
        <!--Zero or more repetitions:-->
        <OrderBy>?</OrderBy>
        <!--Zero or more repetitions:-->
```

```
        <SortBy>?</SortBy>
        <!--Zero or more repetitions:-->
        <CustomerUserID>?</CustomerUserID>
      </TicketSearch>
```

11.7.2. Examples:

11.7.2.1. Web Service Configuration

The following is a basic but complete web service configuration file in YAML format to use all the Ticket Connector operations with the SOAP network transport. In order to use it in OTRS you need to copy the content, save it into a file and call it `GenericTicketConnectorSOAP.yml`, and import it into OTRS in the Web Services screen in the Admin panel by clicking in the "Add web service" button from the overview screen and then clicking in the "Import web service" button in the add screen.

```
---
Debugger:
  DebugThreshold: debug
  TestMode: 0
Description: Ticket Connector SOAP Sample
FrameworkVersion: 3.4.x git
Provider:
  Operation:
    SessionCreate:
      Description: Creates a Session
      MappingInbound: {}
      MappingOutbound: {}
      Type: Session::SessionCreate
    TicketCreate:
      Description: Creates a Ticket
      MappingInbound: {}
      MappingOutbound: {}
      Type: Ticket::TicketCreate
    TicketUpdate:
      Description: Updates a Ticket
      MappingInbound: {}
      MappingOutbound: {}
      Type: Ticket::TicketUpdate
    TicketGet:
      Description: Retrieves Ticket data
      MappingInbound: {}
      MappingOutbound: {}
      Type: Ticket::TicketGet
    TicketSearch:
      Description: Search for Tickets
      MappingInbound: {}
      MappingOutbound: {}
      Type: Ticket::TicketSearch
  Transport:
    Config:
      MaxLength: 100000000
      NameSpace: http://www.otrs.org/TicketConnector/
    Type: HTTP::SOAP
RemoteSystem: ''
Requester:
  Transport:
    Type: ''
```

Similar example can be done for the REST network transport, REST web services uses HTTP operations such as "POST", "GET", "PUT", "PATCH" etc. This operations in conjunction with a URI path called resource defines a OTRS Generic Interface Operation or Invoker (depending on the communication way).

The following example uses /Session resource for SessionCreate, /Ticket resource for TicketSearch and TicketCreate and resource /Ticket/{TicketID} for TicketGet and TicketUpdate (Where {TicketID} is the actual TicketID value of a ticket e.g. /Ticket/123). In order to use it in OTRS you need to copy the content, save it into a file and call it `GenericTicketConnectorREST.yml`, and import it into OTRS in the Web Services screen in the Admin panel by clicking in the "Add web service" button from the overview screen and then clicking in the "Import web service" button in the add screen.

```
---
Debugger:
  DebugThreshold: debug
  TestMode: '0'
Description: Ticket Connector REST Sample
FrameworkVersion: 3.4.x git
Provider:
  Operation:
    SessionCreate:
      Description: Creates a Session
      MappingInbound: {}
      MappingOutbound: {}
      Type: Session::SessionCreate
    TicketCreate:
      Description: Creates a Ticket
      MappingInbound: {}
      MappingOutbound: {}
      Type: Ticket::TicketCreate
    TicketGet:
      Description: Retrieves Ticket data
      MappingInbound: {}
      MappingOutbound: {}
      Type: Ticket::TicketGet
    TicketSearch:
      Description: Search for Tickets
      MappingInbound: {}
      MappingOutbound: {}
      Type: Ticket::TicketSearch
    TicketUpdate:
      Description: Updates a Ticket
      MappingInbound: {}
      MappingOutbound: {}
      Type: Ticket::TicketUpdate
  Transport:
    Config:
      KeepAlive: ''
      MaxLength: '100000000'
      RouteOperationMapping:
        SessionCreate:
          RequestMethod:
          - POST
          Route: /Session
        TicketCreate:
          RequestMethod:
          - POST
          Route: /Ticket
        TicketGet:
          RequestMethod:
          - GET
          Route: /Ticket/:TicketID
        TicketSearch:
          RequestMethod:
          - GET
          Route: /Ticket
        TicketUpdate:
          RequestMethod:
          - PATCH
          Route: /Ticket/:TicketID
    Type: HTTP::REST
RemoteSystem: ''
Requester:
```

```
  Transport:
    Type: ''
```

11.7.2.2. Perl SOAP Requester

The following code is a Perl script that can connect to OTRS via the generic interface. In order to perform the operations provided by the Ticket Connector, it uses two Perl CPAN modules SOAP::Lite and Data::Dumper. Please make sure that your environment is capable to use these modules before you try to run the script.

```
#!/usr/bin/perl -w
# --
# otrs.SOAPRequest.pl - sample to send a SOAP request to OTRS Generic Interface Ticket
 Connector
# Copyright (C) 2001-2016 OTRS AG, http://otrs.com/
# --
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU AFFERO General Public License as published by
# the Free Software Foundation; either version 3 of the License, or
# any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU Affero General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
# or see http://www.gnu.org/licenses/agpl.txt.
# --

use strict;
use warnings;

# use ../ as lib location
use File::Basename;
use FindBin qw($RealBin);
use lib dirname($RealBin);

use SOAP::Lite;
use Data::Dumper;

# ---
# Variables to be defined.

# this is the URL for the web service
# the format is
# <HTTP_TYPE>:://<OTRS_FQDN>/nph-genericinterface.pl/Webservice/<WEB_SERVICE_NAME>
# or
# <HTTP_TYPE>:://<OTRS_FQDN>/nph-genericinterface.pl/WebserviceID/<WEB_SERVICE_ID>
my $URL = 'http://localhost/otrs/nph-genericinterface.pl/Webservice/GenericTicketConnector';

# this name space should match the specified name space in the SOAP transport for the web
 service.
my $NameSpace = 'http://www.otrs.org/TicketConnector/';

# this is operation to execute, it could be TicketCreate, TicketUpdate, TicketGet,
 TicketSearch
# or SessionCreate. and they must to be defined in the web service.
my $Operation = 'TicketCreate';

# this variable is used to store all the parameters to be included on a request in XML
 format. Each
# operation has a determined set of mandatory and non mandatory parameters to work
 correctly. Please
# check the OTRS Admin Manual in order to get a complete list of parameters.
```

```
my $XMLData = '
<UserLogin>some user login</UserLogin>
<Password>some password</Password>
<Ticket>
    <Title>some title</Title>
    <CustomerUser>some customer user login</CustomerUser>
    <Queue>some queue</Queue>
    <State>some state</State>
    <Priority>some priority</Priority>
</Ticket>
<Article>
    <Subject>some subject</Subject>
    <Body>some body</Body>
    <ContentType>text/plain; charset=utf8</ContentType>
</Article>
';

# ---

# create a SOAP::Lite data structure from the provided XML data structure.
my $SOAPData = SOAP::Data
    ->type( 'xml' => $XMLData );

my $SOAPObject = SOAP::Lite
    ->uri($NameSpace)
    ->proxy($URL)
    ->$Operation($SOAPData);

# check for a fault in the soap code.
if ( $SOAPObject->fault ) {
    print $SOAPObject->faultcode, " ", $SOAPObject->faultstring, "\n";
}

# otherwise print the results.
else {

    # get the XML response part from the SOAP message.
    my $XMLResponse = $SOAPObject->context()->transport()->proxy()->http_response()-
>content();

    # deserialize response (convert it into a perl structure).
    my $Deserialized = eval {
        SOAP::Deserializer->deserialize($XMLResponse);
    };

    # remove all the headers and other not needed parts of the SOAP message.
    my $Body = $Deserialized->body();

    # just output relevant data and no the operation name key (like TicketCreateResponse).
    for my $ResponseKey ( keys %{$Body} ) {
        print Dumper( $Body->{$ResponseKey} );
    }
}
```

11.7.2.3. Perl REST Requester

The following code is a Perl script that can connect to OTRS via the generic interface. In order to perform the operations provided by the Ticket Connector, it uses three Perl CPAN modules JSON, REST::Client and Data::Dumper. Please make sure that your environment is capable to use these modules before you try to run the script.

```


#!/usr/bin/perl
# --
# otrs.RESTRequest.pl - sample to send a REST request to OTRS Generic Interface Ticket
 Connector
# Copyright (C) 2001-2016 OTRS AG, http://otrs.com/
# --
```

```
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU AFFERO General Public License as published by
# the Free Software Foundation; either version 3 of the License, or
# any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU Affero General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301 USA
# or see http://www.gnu.org/licenses/agpl.txt.
# --

use strict;
use warnings;

## nofilter(TidyAll::Plugin::OTRS::Perl::Dumper)

# use ../ as lib location
use File::Basename;
use FindBin qw($RealBin);
use lib dirname($RealBin);

use JSON;
use REST::Client;

# ---
# Variables to be defined

# This is the HOST for the web service the format is:
# <HTTP_TYPE>:://<OTRS_FQDN>/nph-genericinterface.pl
my $Host = 'http://localhost/otrs/nph-genericinterface.pl';

my $RestClient = REST::Client->new(
    {
        host => $Host,
    }
);

# This is the Controller and Request the format is:
# /Webservice/<WEB_SERVICE_NAME>/<RESOURCE>/<REQUEST_VALUE>
# or
# /WebserviceID/<WEB_SERVICE_ID>/<RESOURCE>/<REQUEST_VALUE>
# This example will retrieve the Ticket with the TicketID = 1 (<REQUEST_VALUE>)
my $ControllerAndRequest = '/Webservice/GenericTicketConnectorREST/Ticket/1';

my $Params = {
    UserLogin     => "some user login",       # to be filled with valid agent login
    Password      => "some user password",    # to be filled with valid agent password
    DynamicFields => 1,                       # optional, if set to 1,
                                              # ticket dynamic fields included in response
    AllArticles   => 1,                       # optional, if set to 1,
                                              # all ticket articles are included in response
                                              # more options to be found in
         # /Kernel/GenericInterface/Operation/Ticket/TicketGet.pm's
         # Run() subroutine documentation.
};

my @RequestParam;

# As sample web service configuration for TicketGet uses HTTP method GET all other
 parameters needs
# to be sent as URI query parameters

# ----
# For GET method
my $QueryParams = $RestClient->buildQuery( %{ $Params } );

$ControllerAndRequest .= $QueryParams;
```

```
# The @RequestParam array on position 0 holds controller and request
@RequestParam = ($ControllerAndRequest);

$RestClient->GET(@RequestParam);
# ----

# # ----
# # For POST method
# my $JSONParams = encode_json $Params;

# # The @RequestParam array on position 0 holds controller and request
# # on position 1 it holds the JSON data string that gets posted
# @RequestParam = (
#   $ControllerAndRequest,
#   $JSONParams
# );

# $RestClient->POST(@RequestParam);
# # ----

# If the host isn't reachable, wrong configured or couldn't serve the requested page:
my $ResponseCode = $RestClient->responseCode();
if ( $ResponseCode ne '200' ) {
    print "Request failed, response code was: $ResponseCode\n";
    exit;
}

# If the request was answered correctly, we receive a JSON string here.
my $ResponseContent = $RestClient->responseContent();

my $Data = decode_json $ResponseContent;

# Just to print out the returned Data structure:
use Data::Dumper;
print "Response was:\n";
print Dumper($Data);
```

11.7.2.4. cURL Examples for REST Requests

Given the above example on a REST configuration for Generic Ticket Connector we have that:

For Ticket Create: use POST method on /Ticket path.

For Ticket Search: use GET method on /Ticket path.

For Ticket Update: use PATCH method on /Ticket/{TicketID} path (where {TicketID} is a template represented by :TicketID in the transport configuration)

For Ticket Get: use GET method on /Ticket/{TicketID} path (where {TicketID} is a template represented by :TicketID in the transport configuration)

11.7.2.4.1. Create a New Ticket

cURL Command:

```
shell> curl "http://localhost/otrs/nph-genericinterface.pl/Webservice/
GenericTicketConnectorREST/Ticket?UserLogin=agent&Password=123" -H "Content-Type:
 application/json" -d "{\"Ticket\":{\"Title\":\"REST Create Test\", \"Type\": \"Unclassified
\", \"Queue\":\"Raw\",\"State\":\"open\",\"Priority\":\"3 normal\",\"CustomerUser\":
\"customer\"},\"Article\":{\"Subject\":\"Rest Create Test\",\"Body\":\"This is only a test
\",\"ContentType\":\"text/plain; charset=utf8\"}}"  -X POST
```

Response:

```
{
  "ArticleID":5484,
  "TicketNumber":"1001936",
  "TicketID":"1686"
}
```

11.7.2.4.2. Get Ticket Details

cURL Command:

```
curl "http://localhost/otrs/nph-genericinterface.pl/Webservice/GenericTicketConnectorREST/
Ticket/1686?UserLogin=agent&Password=123"
```

Response:

```
{
  "Ticket": [
    {
      "Age": 777,
      "PriorityID": 3,
      "ServiceID": "",
      "Type": "Unclassified",
      "Responsible": "root@localhost",
      "StateID": 4,
      "ResponsibleID": 1,
      "ChangeBy": 2,
      "EscalationTime": 0,
      "Changed": "2014-06-30 19:08:14",
      "OwnerID": 2,
      "RealTillTimeNotUsed": 0,
      "GroupID": 1,
      "Owner": "agent",
      "CustomerID": "OTRS",
      "TypeID": 1,
      "Created": "2014-06-30 19:08:12",
      "Priority": "3 normal",
      "UntilTime": 0,
      "EscalationUpdateTime": 0,
      "QueueID": 2,
      "Queue": "Raw",
      "State": "open",
      "Title": "REST Create Test",
      "CreateBy": 2,
      "TicketID": 1686,
      "StateType": "open",
      "EscalationResponseTime": 0,
      "UnlockTimeout": 0,
      "EscalationSolutionTime": 0,
      "LockID": 1,
      "TicketNumber": "1001936",
      "ArchiveFlag": "n",
      "Lock": "unlock",
      "CreateTimeUnix": 1404173292,
      "SLAID": "",
      "CustomerUserID": "customer"
    }
  ]
}
```

11.7.2.4.3. Update Ticket

cURL Command:

```
curl "http://localhost/otrs/nph-genericinterface.pl/Webservice/GenericTicketConnectorREST/
Ticket/1686?UserLogin=agent&Password=123" -H "Content-Type: application/json" -d "{\"Ticket
\":{\"Queues\":\"Postmaster\"}}"  -X PATCH
```

Response:

```
{
  "TicketNumber":"1001936",
  "TicketID":"1686"
}
```

11.7.2.4.4. Search for Tickets

cURL Command:

```
curl "http://localhost/otrs/nph-genericinterface.pl/Webservice/GenericTicketConnectorREST/
Ticket?UserLogin=agent&Password=123&Queue=Postmaster"
```

Response:

```
{
  "TicketID": [
    "1686",
    "102",
    "100",
    "1"
  ]
}
```

12. The OTRS Daemon

The OTRS Daemon is an independent set of system processes that plan and execute tasks in background, either on a recurrent basis or triggered by events. OTRS Daemon is fundamental for the correct system operation.

In previous versions of OTRS (from 3.1 to 4) there was another process called OTRS Scheduler that does part of the work that the OTRS Demon do in OTRS 5. This old process is replaced by the OTRS Daemon which was re-written from the ground to make it more stable, scalable and robust than its predecessor.

The OTRS Daemon is capable to handle up to 10 tasks at the same time and it can work cooperatively with other OTRS Daemons on different frontend servers in a cluster environment.

When idle OTRS Daemon consist in five processes:

- The main daemon (`bin/otrs.Daemon.pl`)

  This process is in charge to start and keep running the other children daemons.

- Task worker daemon (`Kernel/System/Daemon/DaemonModules/SchedulerTaskWorker.pm`)

  This daemon executes all tasks that have in a list, in a first in first out basis. It can handle simultaneous tasks by creating its own children processes and it checks the task list several times per second. The task list can be filled by task manager daemons, event handlers, and other parts of the system.

  Its main mission is to handle all the tasks in the list as soon as possible.

- Future task manager daemon (`Kernel/System/Daemon/DaemonModules/SchedulerFutureTaskManager.pm`)

  This daemon checks for non recurring tasks that are set to be executed in the future (e.g. when a Generic Interface invoker tries to reach a server and it can't, a task could be set to schedule for execution in the next 5 minutes). At the correct time it sends it the task worker daemon.

- Cron task manager daemon (`Kernel/System/Daemon/DaemonModules/SchedulerCronTaskManager.pm`)

  This daemon calculates when is the next execution time of all recurring tasks (e.g. a cache cleanup one time per week). This kind of tasks are specified in the SysConfig. At the right time for each task it sends the required information to the task worker daemon to execute them.

  ### Note

  If a task execution time definition is changed in SysConfig, it might take up to an hour for the daemon to pick up the change automatically. Alternatively the OTRS Daemon can be restarted to apply the change immediately.

- Generic Agent task manager daemon (`Kernel/System/Daemon/DaemonModules/SchedulerGenericAgentTaskManager.pm`)

  This daemon scans for Generic Agent jobs stored in the database that have a time schedule (discarding all other Generic Agent jobs that are set to executed by events). When is time to run a Generic Agent job it sends the task information to the task worker daemon to handle the task.

  ### Note

  The number of active processes depends on the number of tasks that the OTRS Daemon is executing simultaneously in a time frame.

By default the each daemon logs all error messages on a separated file located in `/opt/otrs/var/log/Daemon/*.log`. These logs are kept in the system for a defined period. To change this behavior and/or to also log the non error messages, please update SysConfig settings in `Daemon` -> `Core::Log`.

When a task could not be executed successfully for any reason, an email is sent to a predefined recipient reporting the issue. The content of the email includes the error messages and trace (if available).

The OTRS Daemon is an automated process that normally does not require human interaction. However it is possible to query its status and start or stop it if needed.

To be sure that the OTRS Daemon is running there is a Cron job that constantly checks that the process is alive. The main daemon is prepared to work even without a database connection, so is perfectly safe if the Cron task to start it is executed even before the database process in the system startup, and it is also tolerant to database disconnections.

If for any reason the OTRS Daemon needs to be stop (for example during a system maintenance), all unhandled tasks are saved, and as soon as the process is started again it continues with all pending tasks. For recurring tasks it will only execute the last instance of the task (if its due time was during the downtime).

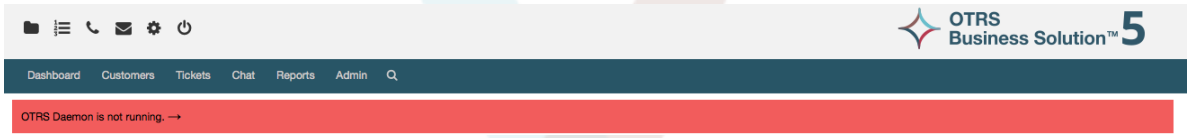
12.1. OTRS Daemon Graphical Interface

The OTRS Daemon is not visible in the OTRS Graphical User Interface unless it stops running.

When the system detects that the OTRS Daemon is not running, a notification is presented to a defined group of users ("admin" by default).

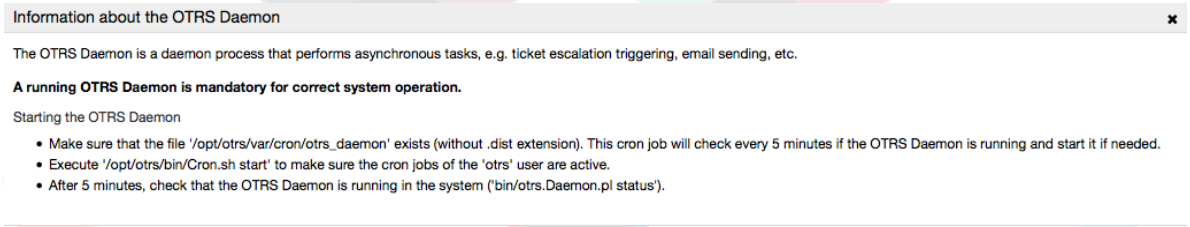
To disable the notification (not recommended), change or add the notification groups, please edit the `Frontend::NotifyModule###800-Daemon-Check` setting in the SysConfig.

Figure 4.114. Daemon notification



Clicking the notification the system presents an overlay window explaining the steps to bring the OTRS Daemon up and running.

Figure 4.115. Start Daemon



12.2. OTRS Daemon Command Line Interface

The OTRS Daemon command line tools let you control the main daemon process (Start / Stop) or query its status. There are also tools to get more detailed information about the other four children daemons.

12.2.1. Main Daemon Tools

To start, stop or query daemon status `bin/otrs.Daemon.pl` script is used.

Example 4.30. Example to start the OTRS Daemon

```
shell> cd /opt/otrs/
shell> OTRS_HOME/bin/otrs.Daemon.pl start
```

Available Options

- **start** - to start the OTRS Daemon process.
- **stop** - to stop the OTRS Daemon process.

- **status** - to query the OTRS Damon process status.
- **start --debug** - to start the OTRS Daemon process in debug mode.

  In this mode each daemon reports different messages depending on the actions that are been executed. This mode is not recommended for production environments.
- **stop --force** - to stop the OTRS Daemon process in reducing the wait for children processes to finish.

  A forced stop reduces the amount of time the main daemon waits to successful stop the other children processes from 30 seconds (normal) to 5 seconds (forced).

12.2.2. Other Daemon Tools

To list all configured child daemons that the main daemon should start and keep running use the console command: `Maint::Daemon::List`.

Example 4.31. Example to list all configured daemons

```
shell> cd /opt/otrs/
shell> bin/otrs.Console.pl Maint::Daemon::List
```

To list detailed information of all daemons use the console command: `Maint::Daemon::Summary`.

Example 4.32. Example to a summary of all daemon tasks

```
shell> cd /opt/otrs/
shell> bin/otrs.Console.pl Maint::Daemon::Summary
```

Chapter 5. Customization

1. Access Control Lists (ACLs)

1.1. Introduction

From OTRS 2.0 on, Access Control Lists (ACLs) can be used to control access to tickets, modules, queues, etc., or to influence actions on tickets (closing, moving, etc.) in certain situations. ACLs can be used to supplement the existing permission system of roles and groups. Using ACLs, rudimentary work-flows within the system can be mapped, based on ticket attributes.

In a general way ACLs are used to reduce the possible options for a ticket based on a defined set of rules.

ACLs can be directly entered into the `Kernel/Config.pm` file. However this is not any more recommended as OTRS comes now with a GUI Access Control Lists in the Admin panel that allows to save the ACLs in the Database as the first step and then deploy them into a file when they are ready.

This chapter has some ACL examples which will walk you through the process of defining ACL definitions, and a reference of all possible important ACL settings.

Warning

The default user 'root@localhost' is not affected by the Ticket ACLs

1.2. Definition

The ACL definition can be split into two big parts, 'Matching' and 'Change'. In the matching sections the ACLs contains attributes that has to be met in order to use the ACL. If the attributes defined in the ACL does not match with the attributes that are sent, then the ACL does not take any affect, but any other match ACL will. The change sections contains the rules to reduce the possible options for a ticket.

Matching Sections

- Properties

  This section contains matching options that can be changed on the fly. For example on a ticket creation time the data of the ticket changes dynamically as the agent sets the information. If an ACL is set to match a ticket attribute then only when the matching attribute is selected the ACL will be active and might reduce other ticket attributes, but as soon as another value is selected the ACL will not take any affect.

- PropertiesDatabase

  This section is similar to 'Properties' but does not take changes in ticket attributes that are not saved into the DataBase, this means that changing an attribute without submit will not make any effect. This section is not use for ticket creation screens (as tickets are not yet created in the Database).

Change Sections

- Possible

  Possible section resets the data to be reduce to only the elements that are set in this section.

- PossibleAdd

  Elements in PossibleAdd section add missing elements that were reduced in other ACLs. PossibleAdd is only used in together with other ACLs that have Possible or PossibleNot sections.

- PossibleNot

  This section is used to remove specific elements from the current data. It could be used stand alone or together with other ACLs with a Possible or PossibleAdd sections.

In order to make the development of ACLs easier and more powerful there is a set of so called modifiers for the attributes on each section. This modifiers are explained below:

Modifiers

- [Not]

  This modifier is used to negate a value for example: '[Not]2 low' in this case talking about ticket priorities will be the same as to have: '1 very low', '3 normal', '4 high', '5 very high'.

- [RegExp]

  It is use to define a regular expression for matching several values, for example '[RegExp]low' talking about priorities is the same as '1 very low', '2 low'.

- [regexp]

  It is very similar to [RegExp] but it is case insensitive.

- [NotRegExp]

  Negated regular expressions for example '[NotRegExp]low' talking about priorities is the same as '3 normal', '4 high', '5 very high'.

- [Notregexp]

  It is very similar to [NotRegExp] but it is case insensitive.

1.3. Examples

The following examples are shown in both ways graphical and text based.

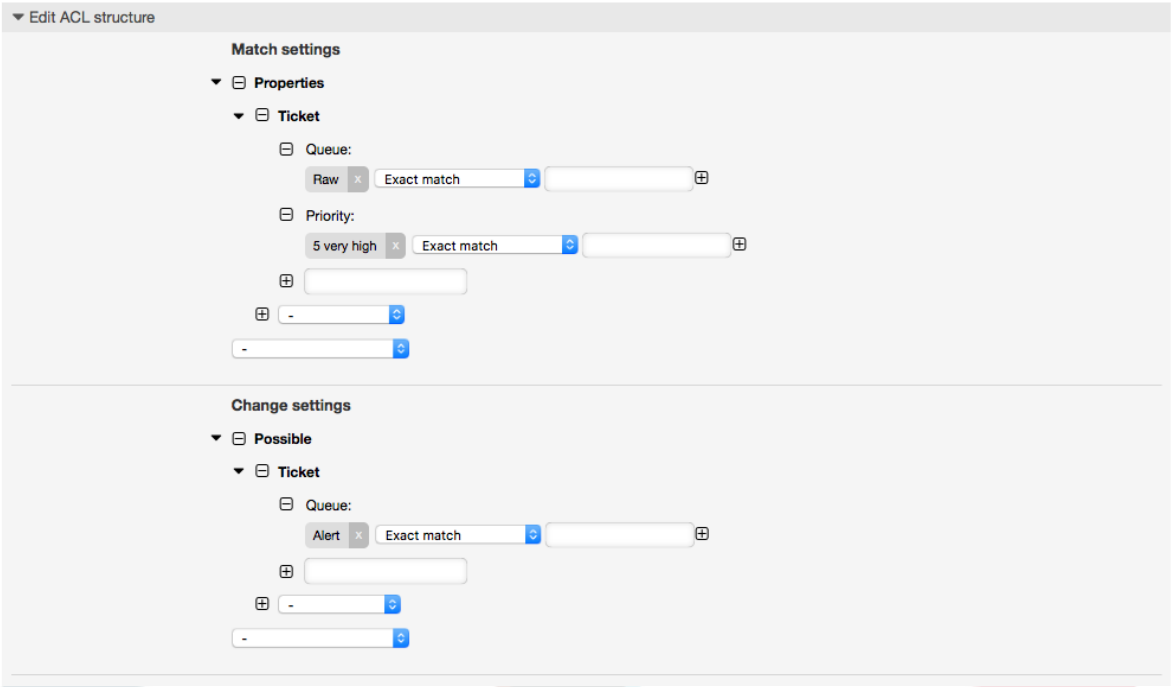
Example 5.1. ACL allowing movement into a queue of only those tickets with ticket priority 5.

This example shows you the basic structure of an ACL. First, it needs to have a name. In this case, it is "100-Example-ACL". Note that the ACLs will be numerically sorted before execution, so you should use the names carefully.

Secondly, you have a "Properties" section which is a filter for your tickets. All the criteria defined here will be applied to a ticket to determine if the ACL must be applied or not. In our example, a ticket will match if it is in the queue "Raw" and has priority "5 very high". This is also affected by changes in the form (e.g. if the ticket is the queue "Raw" and had a priority "3 normal" at this moment the ACL will not match, but then priority drop-down is selected and the priority is changed now to "5 very high" then will also match).

Lastly, a section "Possible" defines modifications to the screens. In this case, from the available queues, only the queue "Alert" can be selected in a ticket screen.

Figure 5.1. ACL 100-Example-ACL

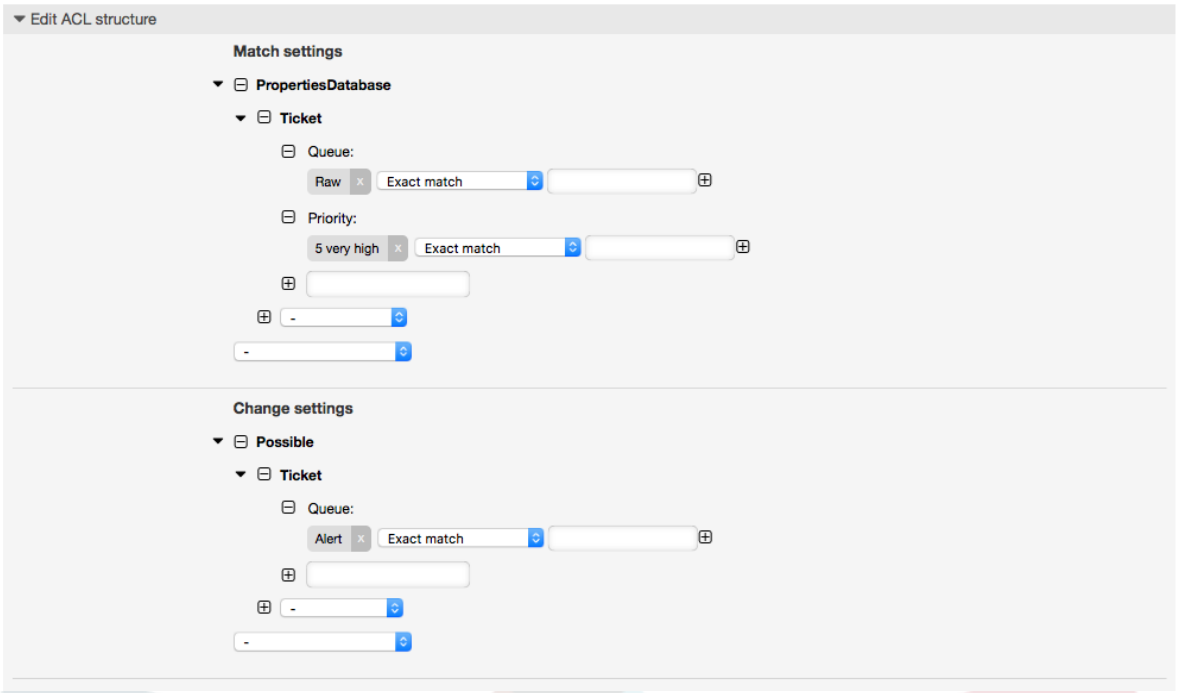



```
# ticket acl
$Self->{TicketAcl}->{'100-Example-ACL'} = {
    # match properties
    Properties => {
        # current ticket match properties
        Ticket => {
            Queue => ['Raw'],
            Priority => ['5 very high'],
        }
    },
    # return possible options (white list)
    Possible => {
        # possible ticket options (white list)
        Ticket => {
            Queue => ['Alert'],
        },
    },
};
```

Example 5.2. ACL allowing movement into a queue of only those tickets with ticket priority 5 stored in the database.

This example is very similar to the last one, but in this case only tickets in the queue "Raw" and with a priority "5 very high", both stored in the database will match. This kind of ACLs does not consider changes in the form before the ticket is really updated in the database.

Figure 5.2. ACL 102-Example-ACL

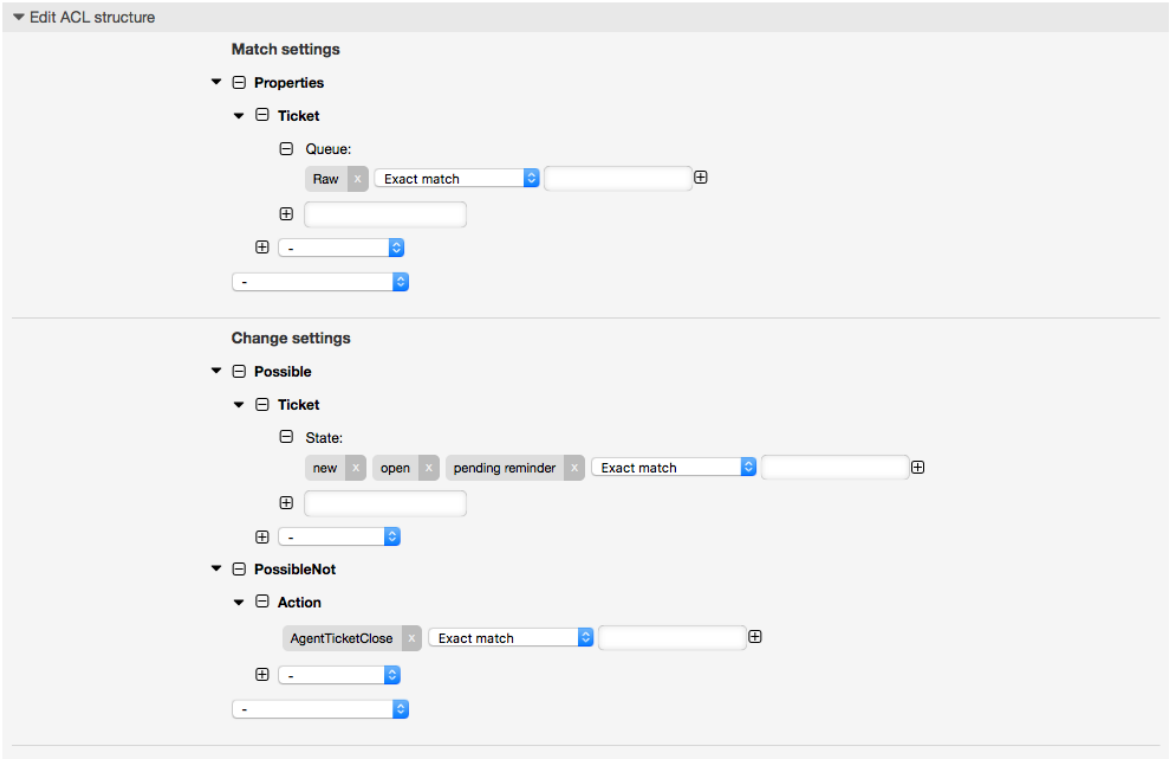



```
# ticket acl
$Self->{TicketAcl}->{'102-Example-ACL'} = {
    # match properties
    PropertiesDatabase => {
        # current ticket match properties
        Ticket => {
            Queue => ['Raw'],
            Priority => ['5 very high'],
        }
    },
    # return possible options (white list)
    Possible => {
        # possible ticket options (white list)
        Ticket => {
            Queue => ['Alert'],
        },
    },
};
```

Example 5.3. ACL disabling the closing of tickets in the raw queue, and hiding the close button.

Here you can see how a ticket field (state) can be filtered with more than one possible value to select from. It is also possible to limit the actions that can be executed for a certain ticket. In this case, the ticket cannot be closed.

Figure 5.3. ACL 102-Second-Example-ACL

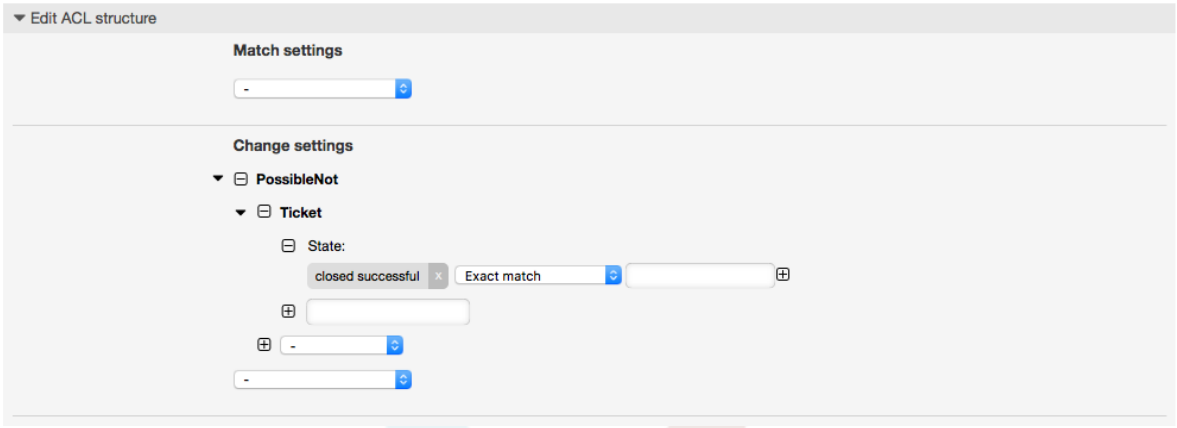



```
$Self->{TicketAcl}->{'102-Second-Example-ACL'} = {
    # match properties
    Properties => {
        # current ticket match properties
        Ticket => {
            Queue => ['Raw'],
        }
    },
    # return possible options (white list)
    Possible => {
        # possible ticket options (white list)
        Ticket => {
            State => ['new', 'open', 'pending reminder'],
        },
    },
    # return also not possible options (black list)
    PossibleNot => {
        # not possible action options
        Action => [ 'AgentTicketClose' ],
    },
};
```

Example 5.4. ACL removing always state closed successful.

This example shows how it is possible to define negative filters (the state "closed successful" will be removed). You can also see that not defining match properties for a ticket will match any ticket, i. e. the ACL will always be applied. This may be useful if you want to hide certain values by default, and only enable them in special circumstances (e. g. if the agent is in a specific group).

Figure 5.4. ACL 103-Third-ACL-Example

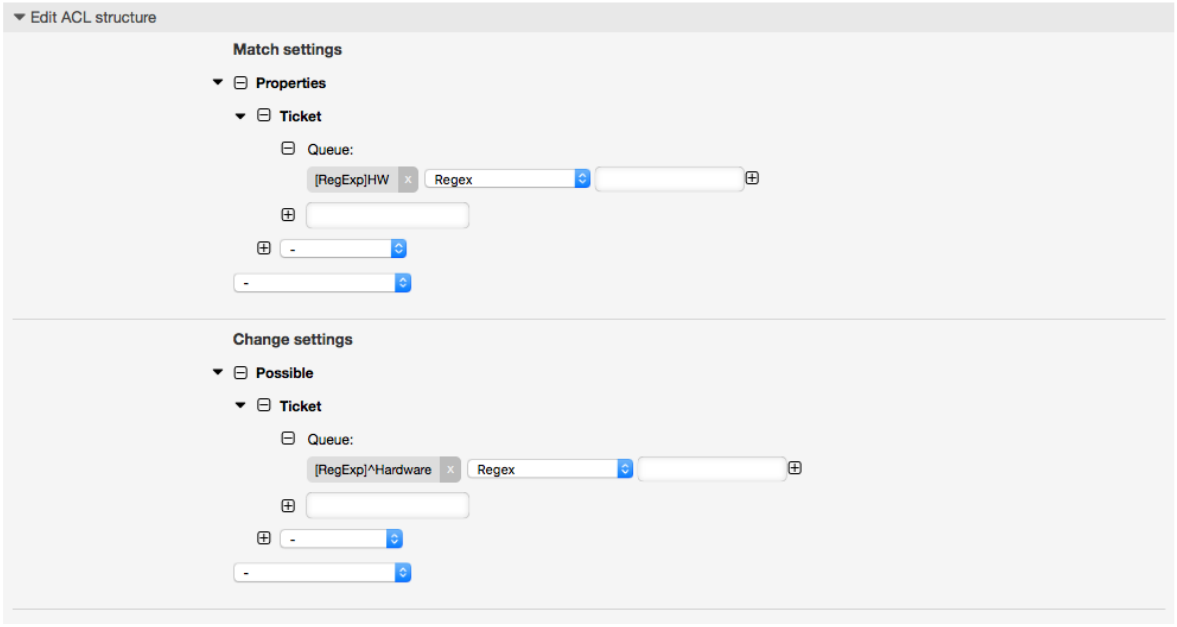



```
$Self->{TicketAcl}->{'103-Third-ACL-Example'} = {
    # match properties
    Properties => {
        # current ticket match properties (match always)
    },
    # return possible options
    PossibleNot => {
        # possible ticket options
        Ticket => {
            State => ['closed successful'],
        },
    },
};
```

Example 5.5. ACL only showing Hardware services for tickets that are created in queues that start with "HW".

This example also shows you how you can use regular expressions for matching tickets and for filtering the available options.

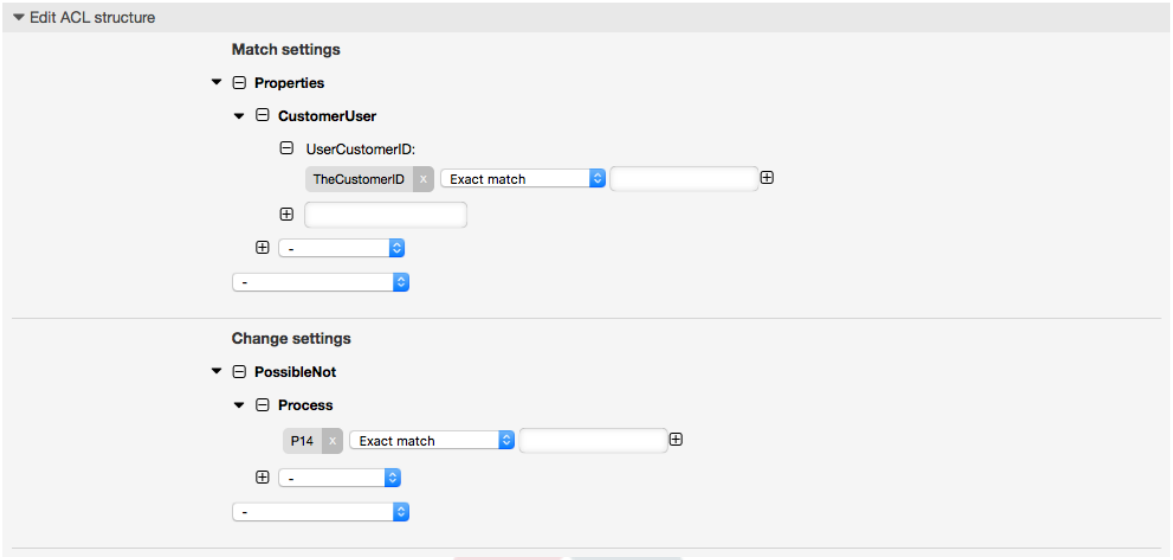
Figure 5.5. ACL 104-Only-Hardware-Services-for-HW-Queues-ACL



```
$Self->{TicketAcl}->{'104-Only-Hardware-Services-for-HW-Queues-ACL'} = {
    # match properties
    # note we don't have "Ticket => {" because there's no ticket yet
    Properties => {
    Queue => {
        Name => ['[RegExp]HW'],
        }
    },
    # return possible options
    Possible => {
        # possible ticket options
        Ticket => {
            Service => ['[RegExp]^(Hardware)'],
        },
    },
};
```

Example 5.6. ACL to restrict a Process in the customer frontend using the CustomerID.

Figure 5.6. ACL 105-Disallow-Process-For-CustomerID




```
$Self->{TicketAcl}->{"105-Disallow-Process-For-CustomerID"} = {
    'Possible' => {},
    'PossibleNot' => {
        'Process' => [
            'P14'
        ]
    },
    'Properties' => {
        'CustomerUser' => {
            'UserCustomerID' => [
                'CustomerID'
            ]
        }
    },
    'PropertiesDatabase' => {},
    'StopAfterMatch' => 0
};
```

1.4. Reference

In the example below there is a list of all parameters which can be used for ACLs.

Please see the section on ACLs in the ProcessManagement documentation for a detailed description of how to use ACLs for process tickets.

Example 5.7. Reference showing all possible important ACL settings.

```
# ticket acl
$Self->{TicketAcl}->{'200-ACL-Reference'} = {
    # match properties (current values from the form)
    Properties => {

        # the used frontend module
        Frontend => {
            Action => ['AgentTicketPhone', 'AgentTicketEmail'],
        },

        # the logged in agent
        User => {
            UserLogin => ['some login'],
            Group_rw => [
                'hotline',
            ],
            Role => [
                'admin',
            ],
            # ...
        },

        # the logged in customer
        CustomerUser => {
            UserLogin => ['some login'],
            UserCustomerID => ['some customer id'],
            Group_rw => [
                'hotline',
            ],
            Role => [
                'admin',
            ],
            # ...
        },

        # process properties
        Process => {
            ProcessEntityID        => ['Process-9c378d7cc59f0fce4cee7bb9995ee3eb'],
 # the Process that the current ticket is part of
            ActivityEntityID       => ['Activity-f8b2fdebe54eeb7b147a5f8e1da5e35c'],
 # the current Activity of the ticket
            ActivityDialogEntityID => ['ActivityDialog-aff0ae05fe6803f38de8fff6cf33b7ce'],
 # the current ActivityDialog that the Agent/Customer is using
        },

        # ticket properties
        Queue => {
            Name     => ['Raw'],
            QueueID  => ['some id'],
            GroupID  => ['some id'],
            Email    => ['some email'],
            RealName => ['OTRS System'],
            # ...
        },
        Service => {
            ServiceID => ['some id'],
            Name      => ['some name'],
            ParentID  => ['some id'],
            # ...
        },
        Type => {
            ID   => ['some id'],
            Name => ['some name'],
            # ...
```

```
        },
        Priority = {
            ID   => ['some id'],
            Name => ['some name'],
            # ...
        },
        SLA = {
            SLAID    => ['some id'],
            Name     => ['some name'],
            Calendar => ['some calendar'],
            # ...
        },
        State = {
            ID       => ['some id'],
            Name     => ['some name'],
            TypeName => ['some state type name'],,
            TypeID   => ['some state type id'],
            # ...
        },
        Owner => {
            UserLogin => ['some login'],
            Group_rw => [
                'some group',
            ],
            Role => [
                'admin',
            ],
            # ...
        },
        Responsible => {
            UserLogin => ['some login'],
            Group_rw => [
                'some group',
            ],
            Role => [
                'admin',
            ],
            # ...
        },
        DynamicField => {
            # Names must be in DynamicField_<field_name> format.
            # Values in [ ... ] must always be the untranslated internal data keys
            #   specified in the dynamic field definition and
            #   not the data values shown to the user.
            DynamicField_Field1          => ['some value'],
            DynamicField_OtherField      => ['some value'],
            DynamicField_TicketFreeText2 => ['some value'],
            # ...
        },
        # alternatively, ticket properties can be specified in the ticket hash
        Ticket => {
            Queue                => ['Raw'],
            State                => ['new', 'open'],
            Priority             => ['some priority'],
            Lock                 => ['lock'],
            CustomerID           => ['some id'],
            CustomerUserID       => ['some id'],
            Owner                => ['some owner'],
            DynamicField_Field1  => ['some value'],
            DynamicField_MyField => ['some value'],
            # ...
        },
    },

    # match properties (existing values from the database)
    PropertiesDatabase => {
        # See section "Properties", the same config can be used here.
        # ...
    }

    # reset possible options (white list)
    Possible => {
```

```
        # possible ticket options (white list)
        Ticket => {
            Queue => ['Hotline', 'Coordination'],
            State => ['some state'],
            Priority => ['5 very high'],
            DynamicField_Field1  => ['some value'],
            DynamicField_MyField => ['some value'],
            # ...
            NewOwner => ['some owner'],
            OldOwner => ['some owner'],
            # ...
        },

        # Limit the number of possible ActivityDialogs the Agent/Customer
        #   can use in a process ticket.
        ActivityDialog => ['AD1', 'AD3'],

        # Limit the number of possible Processes that can be started
        Process => ['Process-9c378d7cc59f0fce4cee7bb9995ee3eb',
 'Process-12345678901234567890123456789012'],

        # possible action options (white list)
        Action => [
            'AgentTicketBounce',
            'AgentTicketPhone'.      # only used to show/hide the Split action
            'AgentLinkObject',       # only used to show/hide the Link action
            # ...
        ],
    },
    # add options (white list)
    PossibleAdd => {
        # See section "Possible"
        # ...
    },
    # remove options (black list)
    PossibleNot => {
        # See section "Possible"
        # ...
    },
};
```

Note

While matching ACLs if CustomerUserID parameter sent, the ACL mechanism will compare the defined ACLs using the supplied CustomerUserID to gather the CustomerUser details to fill the CustomerUser hash and it also overrides the Customer information in the Ticket hash for the Properties match. On the other hand this calculations are also made for the PropertiesDatabase part, but using the Ticket Customer as the basis to gather the data.

Notice that in Customer Interface, the CustomerUserID is always sent with the current logged Customer User.

Be aware that in ticket search screens (AgentTicketSearch and CustomerTicketSearch) the only affected attributes by ACLs are the Dynamic Fields. This means that this screens you can not restrict any other attribute like ticket type, state, queue, etc.

From OTRS 4 the 'Action' parameter is not longer a hash but an array reference and it can be used in any of the Change sections using any of the Modifiers.

2. Process Management

2.1. Introduction

This feature of OTRS allows you to model processes (work-flows) in the ticket system. The basic idea is to be able to define recurring processes, and to delegate work items to different people, as well as leading the progress of a process in different directions based on certain criteria.

2.2. Example process

Let's see an example to make it more demonstrative. We will define a book order process:

2.2.1. Recording the demand

Before an order will be placed, the demand for literature by an employee will be recorded. The following book is needed in our example:

```
Title: Prozessmanagement für Dummies
Autor: Thilo Knuppertz
ISBN: 3527703713
```

2.2.2. Approval by manager

The head of the employee's department needs to decide on the order. In case of a denial, a reason should be recorded by the manager. In case of approval, the order is passed to the purchasing department.

2.2.3. Processing by purchasing department

Purchasing now has the task to find out where the book can be ordered with the best conditions. If it is out of stock, this can be recorded in the order. In case of a successful order purchasing will record the supplier, the price and the delivery date.

2.2.4. Processing by the mail room

The shipment will arrive at the company. The incoming goods department checks the shipment and records the date of receipt. Now the employee will be informed that their order has arrived and is ready to be collected.

2.3. Implementing the example

If we assume that a ticket acts in this work-flow like an accompanying document that can receive change notes, we already have a clear picture of process tickets.

From the analysis of the example process we can identify the following necessary items:

- Possibilities to record data, let's call them *Activity Dialogs*,
- Checks which can react to changed data automatically, let's call them *Transitions*,
- changes which can be applied to a process ticket after successful transitions of a process ticket, let's call them *Transition Actions*.

We also need an additional item which might not be as obvious:

- A possibility to offer more than just one Activity Dialog to be available. In our example this is needed when the manager must have the choice between "Approve" and "Deny". Let's call this *Activity*.

Now, with Activities, Activity Dialogs, Transitions and Transition Actions we have the necessary tools to model the individual steps of our example. What's still missing is an area where for each work-flow the order of the steps can be specified. Let's call this *Process*. To be able to refer to all these entities later, we will assign to them an abbreviation in parentheses. This abbreviation is based on an internal identification mechanism called EntityIDs.

The EntityIDs are conformed with one or two letters (depending on the process part or entity) and then a consecutive number, examples:

- Process: 'P1', 'P2' ... 'Pn'.
- Activity: 'A1', 'A2' ... 'An'.
- Activity Dialog: 'AD1', 'AD2' ... 'ADn'.
- Transition: 'T1', 'T2' ... 'Tn'.
- Transition Action: 'TA1', 'TA2' ... 'TAn'.

Before the creation of the process and its parts is necessary to prepare the system, we will need to define some Queues, Users and Dynamic Fields as well as set some SysConfig options.

Create the following Queues:

- Management
- Employees
- Purchasing
- Post office

Create the following Users:

- Manager
- Employee

Create the following Dynamic Fields:

- Title

Label	Title
Type	Text
Object	Ticket

- Author

Label	Author
Type	Text

Object	Ticket

- ISBN

Label	ISBN
Type	Text
Object	Ticket

- Status

Label	Status
Type	Dropdown
Object	Ticket
Possible Values	• Approval • Approval denied • Approved • Order denied • Order placed • Shipment received

Note: Please use this exactly this possible values for "Key" and "Value" in the Dynamic Field setup.

- Supplier

Label	Supplier
Type	Text
Object	Ticket

- Price

Label	Price
Type	Text
Object	Ticket

- DeliveryDate

Label	Delivery date
Type	Date
Object	Ticket

- DateOfReceipt

Label	Date Of Receipt
Type	Date
Object	Ticket

Set the the following SysConfig settings:

- 'Ticket::Responsible': Yes
- 'Ticket::Frontend::AgentTicketZoom###ProcessWidgetDynamicFieldGroups':

Key:	***Content:***
Book	Title, Author, ISBN
General	Status
Order	Price, Supplier, DeliveryDate
Shipment	DateOfReceipt

- 'Ticket::Frontend::AgentTicketZoom###ProcessWidgetDynamicField':

Key:	***Content:***
Author	1
DateOfReceipt	1
DeliveryDate	1
ISBN	1
Price	1
Status	1
Supplier	1
Title	1

Now lets start with the real Process Management stuff. In the next step, we will define the individual entities that we need.

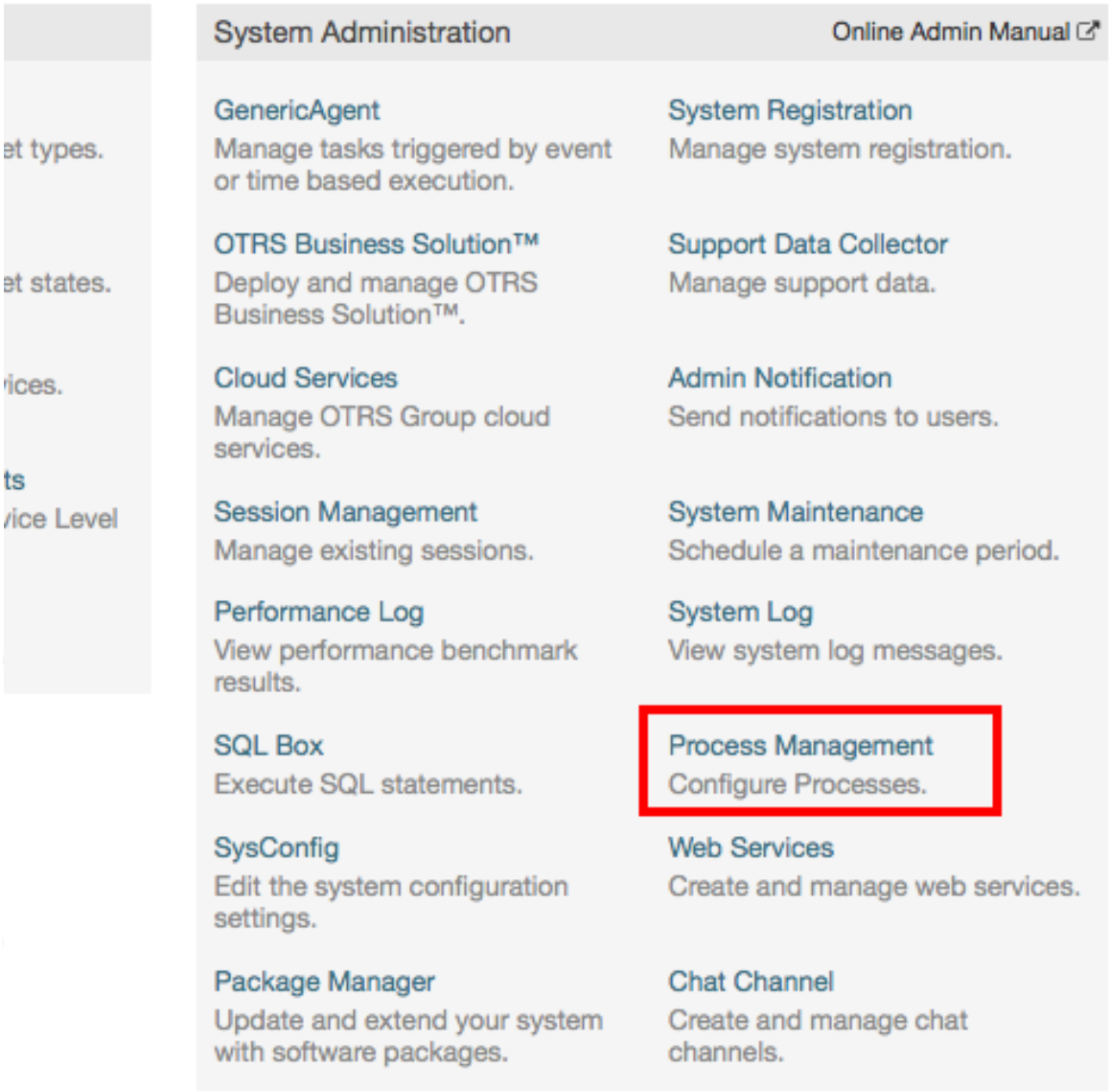
2.3.1. Process (as a container)

To create a new process is necessary to click on the "Process Management" link in the System Administration box in the Admin panel, this will lead to the Process Management Overview screen. After the creation of the process we can create all other entities (or process parts).

Note

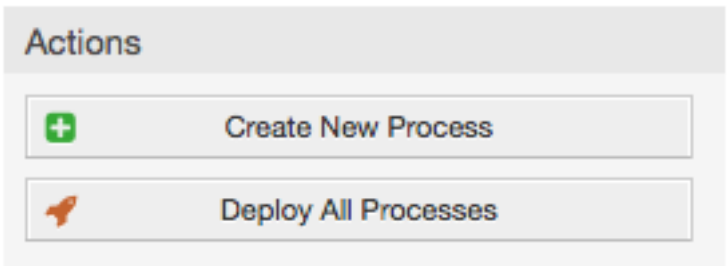
Activities, Activity Dialogs, Transitions and Transition Actions defined in one process will be available for all the processes in the system.

Figure 5.7. OTRS Admin screen - System Administration



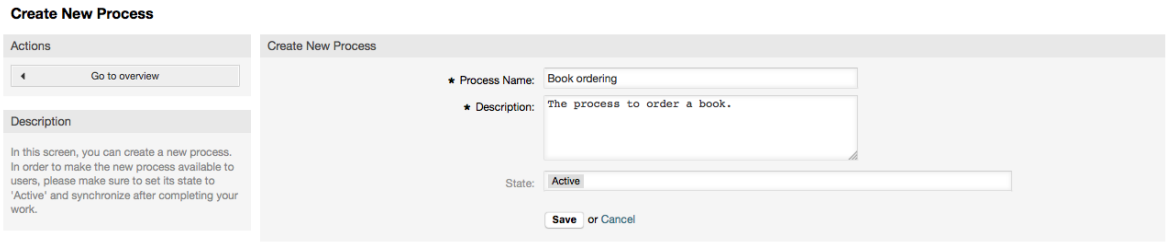
Click on the "Create New Process" action from the Actions box.

Figure 5.8. Create New Process button



Fill the process information,set Process Name and the Description, we will leave the process State as "inactive", until we finish all the tasks. Save the process.

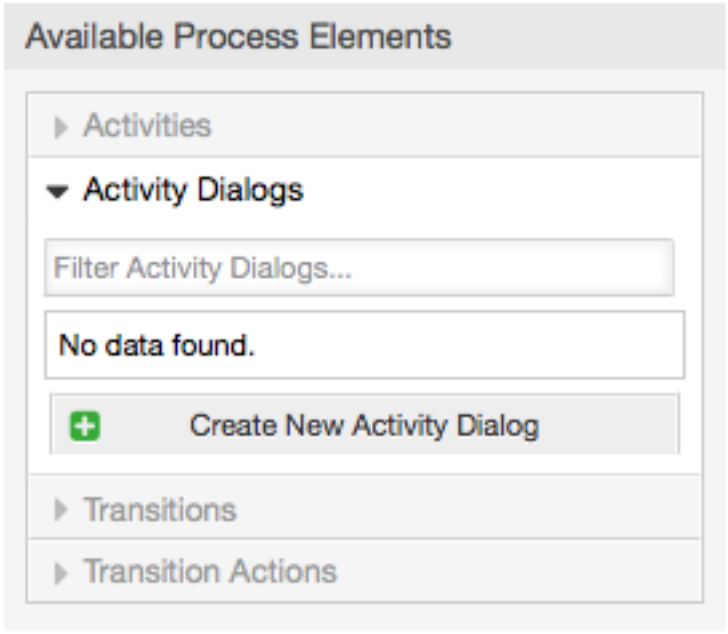
Figure 5.9. Add new process



2.3.2. Activity Dialogs

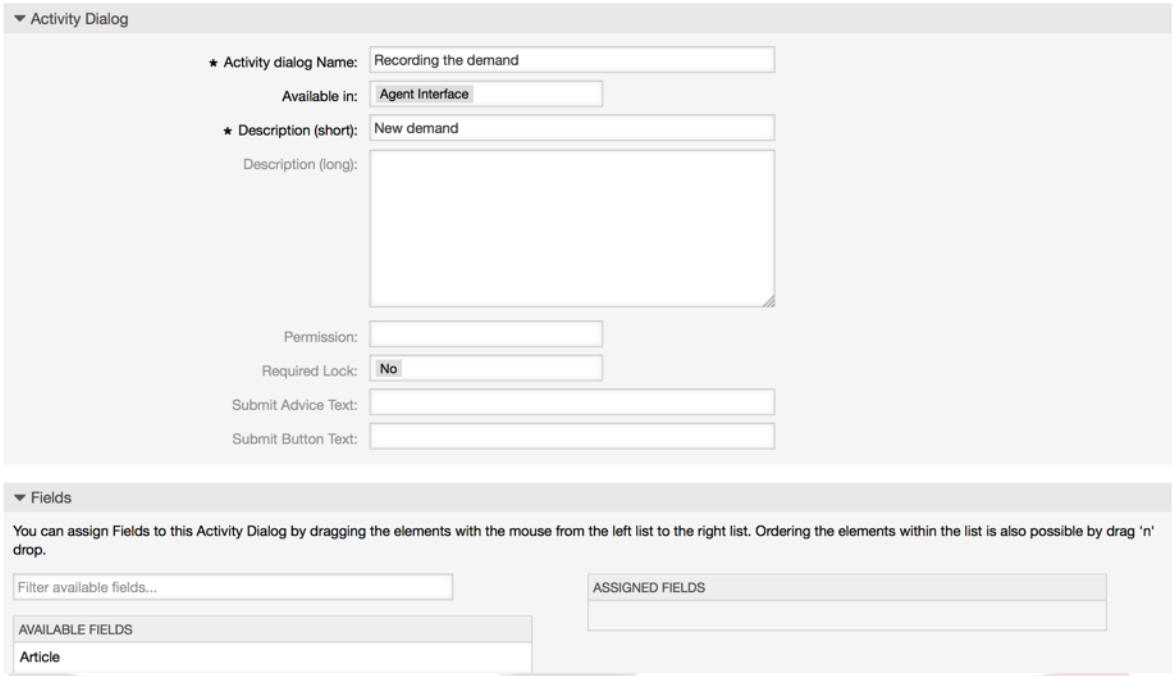
Click on the new process name in the Process Management Overview Screen, then in the "Available Process Elements" click in "Activity Dialogs" (this action will expand the activity dialog options and will collapse all others doing an accordion like effect), then click on "Create New Activity Dialog".

Figure 5.10. Create New Activity Dialog button



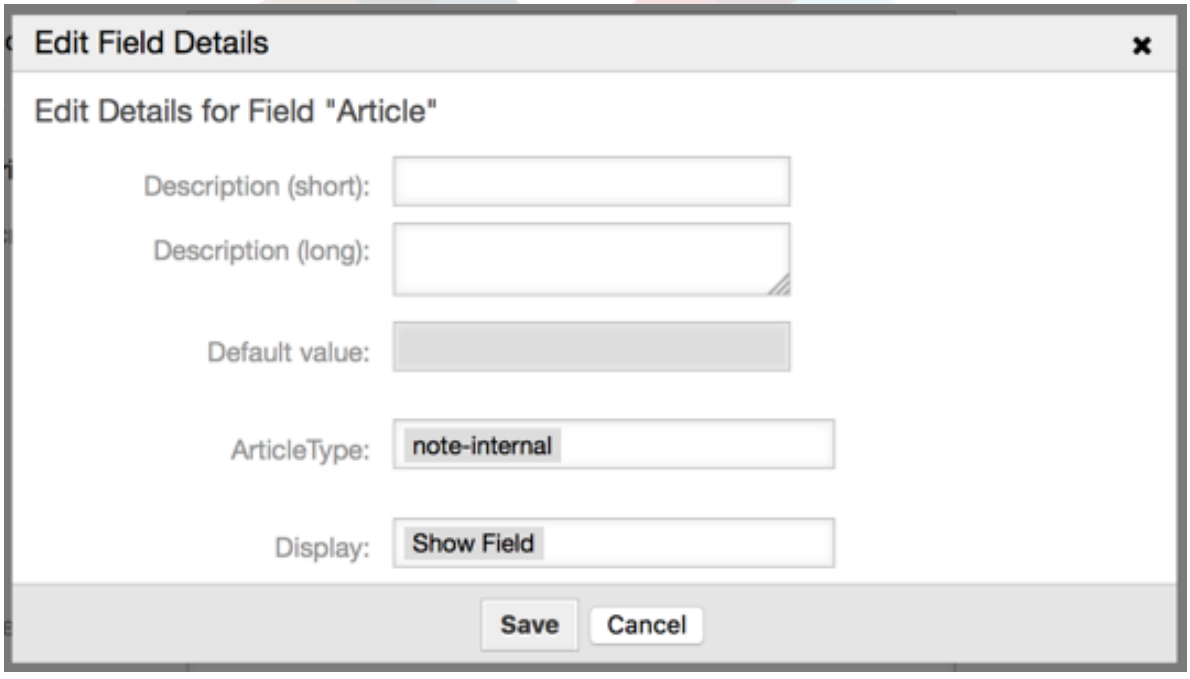
In the opened popup screen fill the "Activity dialog Name" as well as the "Description (short)" fields, for this example we will leave all other fields as the default, to assign fields to the Activity Dialog simple drag the required field from the "Available Fields" pool and drop into the "Assigned Fields" pool. The order in the "Assigned Fields" pool is the order as the fields will have in the screen, to modify the order simply drag and drop the field within the pool to rearrange it in the correct place.

Figure 5.11. Add new Activity Dialog



As soon as the fields are dropped into the "Assigned Fields" pool another popup screen is shown with some details about the field, we will leave the default options and only for Article fields we should make sure that the ArticleType field is set to "note-internal".

Figure 5.12. Edit field details (Article)



After all fields are assigned click on the submit button in the main popup screen to save the changes.

In this example we will use Article field for comments, but another option could be to create a TextArea type Dynamic Field, the rest of the mentioned fields in the lines below are the Dynamic Fields that we define before.

Please be aware that in this screen all the Dynamic Fields has the prefix "DynamicField_" as in "DynamicField_Title". Do not confuse with the field "Title" that is the Ticket Title.

Create the following Activity Dialogs:

- "Recoding the demand" (AD1)

  An Activity Dialog that contains all the required fields for the data to be collected for the order (Title, Author and ISBN), and a Status field with the possibility to choose "Approval".

- "Approval denied" (AD2)

  An Activity Dialog with a comment field (Article) and a Status field with the option "Approval denied".

- "Approved" (AD3)

  Here we just need the Status field with the option "Approved".

- "Order denied" (AD4)

  An activity dialog which makes it possible for purchasing to reject an impossible order (book out of stock). Here we also need a comment field and the Status field with the option "Order denied".

- "Order placed" (AD5)

  An activity dialog with the fields Supplier, Price and Delivery date for purchasing and the Status field with the option "Order placed".

- "Shipment received" (AD6)

  An activity for the mail room with a field for the Date of receipt and the Status field with the option "Shipment received".

To restrict the Status field for each activity dialog we need to add some ACLs in the `Kernel/Config.pm` or to a new Perl file located in `Kernel/Config/Files`.

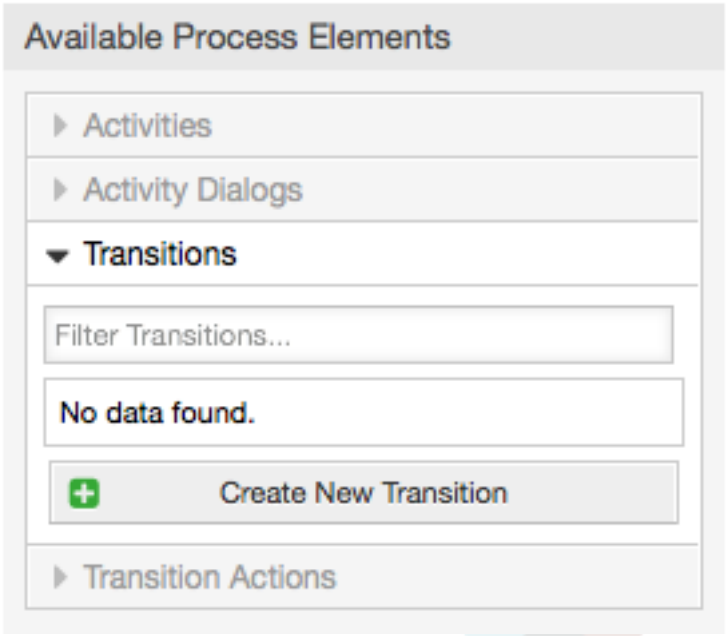
```
$Self->{TicketAcl}->{'P1-AD1-1'} = {
    Properties => {
        Process => {
            ActivityDialogEntityID => ['AD1'],
        },
    },
    Possible => {
        Ticket => {
            DynamicField_Status => ['Approval'],
        },
    },
};

$Self->{TicketAcl}->{'P1-AD2-1'} = {
    Properties => {
        Process => {
            ActivityDialogEntityID => ['AD2'],
        },
    },
    Possible => {
        Ticket => {
            DynamicField_Status => ['Approval denied'],
        },
    },
};
```

```
$Self->{TicketAcl}->{'P1-AD3-1'} = {
    Properties => {
        Process => {
            ActivityDialogEntityID => ['AD3'],
        },
    },
    Possible => {
        Ticket => {
            DynamicField_Status => ['Approved'],
        },
    },
};

$Self->{TicketAcl}->{'P1-AD4-1'} = {
    Properties => {
        Process => {
            ActivityDialogEntityID => ['AD4'],
        },
    },
    Possible => {
        Ticket => {
            DynamicField_Status => ['Order denied'],
        },
    },
};

$Self->{TicketAcl}->{'P1-AD5-1'} = {
    Properties => {
        Process => {
            ActivityDialogEntityID => ['AD5'],
        },
    },
    Possible => {
        Ticket => {
            DynamicField_Status => ['Order placed'],
        },
    },
};

$Self->{TicketAcl}->{'P1-AD6-1'} = {
    Properties => {
        Process => {
            ActivityDialogEntityID => ['AD6'],
        },
    },
    Possible => {
        Ticket => {
            DynamicField_Status => ['Shipment received'],
        },
    },
};
```

2.3.3. Transitions

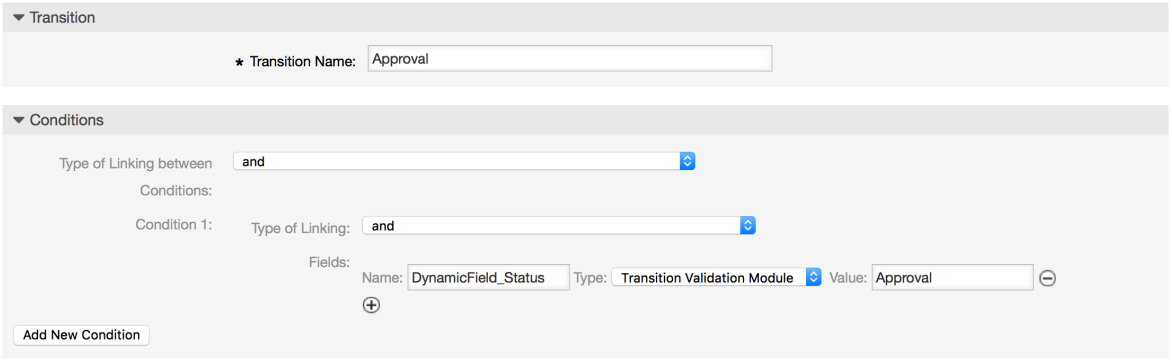
In the "Available Process Elements" click in "Transitions", then click on "Create New Transition".

Figure 5.13. Create New Transition button



In the opened popup screen fill the "Transition Name", then in the conditions, for this examples we will use just one condition and just one field, for both we can leave the Type of Linking as "and" and we will use the filed match type value as "String".

Figure 5.14. Add new Transition



After all conditions are set click on the submit button to save the changes.

Create the following Transitions:

- "Approval" (T1)

  A transition which checks if the Status field is set to "Approval".

- "Approval denied" (T2)

  A transition which checks if the Status field is set to "Approval denied".

- "Approved" (T3)

  A transition which checks if the Status field is set to "Approved".

- "Order denied" (T4)

  A transition which checks if the Status field is set to "Order denied".

- "Order placed" (T5)

  A transition which checks if the Status field is set to "Order placed".

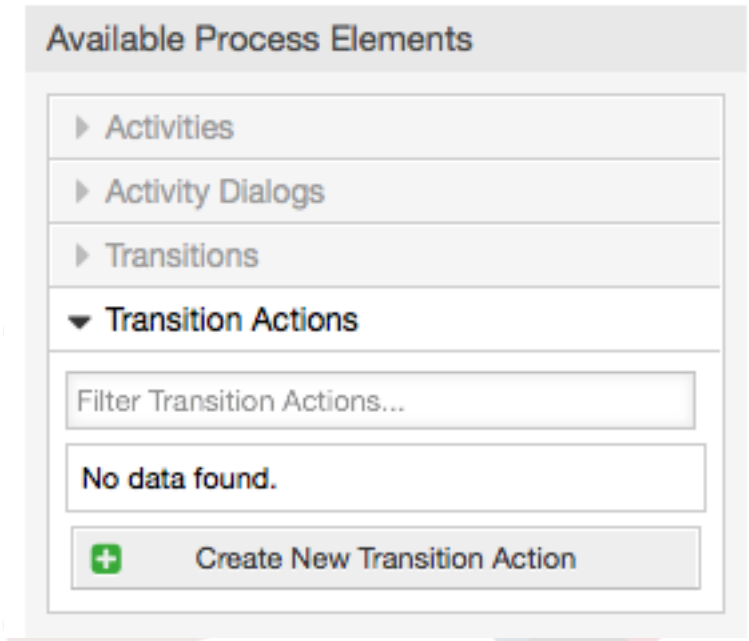
- "Shipment received" (T6)

  A transition which checks if the Status field is set to "Shipment received".

2.3.4. Transition Actions

Click on "Transition Actions" in the "Available Process Elements", then click on "Create New Transition Action".

Figure 5.15. Create New Transition Action button



In the opened popup screen fill the "Transition Action Name", and the "Transition Action module" then add the required and optional parameter names and values.

All the Transition Action Modules are located in `Kernel/System/ProcessManagement/TransitionAction` and the following is the list of bundled Transition Actions included in this release:

- DynamicFieldSet
- TicketArticleCreate
- TicketCreate
- TicketCustomerSet
- TicketLockSet
- TicketOwnerSet
- TicketQueueSet
- TicketResponsibleSet
- TicketServiceSet
- TicketSLASet

- TicketStateSet
- TicketTitleSet
- TicketTypeSet

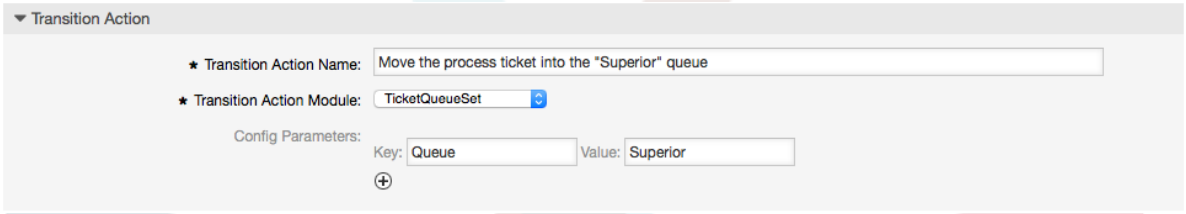
Each module has its own and different parameters. Please review the module documentation to learn all require and optional parameters.

Note

From OTRS 4.0.1, parameters are not longer mandatory fixed values, but instead, they can inherit from the original ticket using format: <OTRS_Ticket_property>.

From OTRS 4.0.6, the format <OTRS_TICKET_property> is now supported, older format is still usable, but deprecated as it will be dropped in further versions.

Figure 5.16. Add new Transition Action



After all parameters and values are set click on the submit button to save the changes.

Create the following Transitions Actions:

- "Move the process ticket into the 'Management' queue" (TA1)

  This action is supposed to be executed when the Transition "Approval" (T1) applied.
- "Change ticket responsible to 'manager'" (TA2)

  To be executed when the Transition "Approval" (T1) applied.
- "Move process ticket into the 'Employees' queue" (TA3)

  To be executed when:
  - The Transition "Approval denied" (T2) applied
  - The Transition "Order denied" (T4) applied
  - The Transition "Shipment received" (T6) applied
- "Change ticket responsible to 'Employee'" (TA4)

  To be executed when:
  - The Transition "Approval denied" (T2) applied
  - The Transition "Order denied" (T4) applied
  - The Transition "Shipment received" (T6) applied
- "Move process ticket into the 'Purchasing' queue" (TA5)

  To be executed when the transition "Approved" (T3) applied.
- "Move process ticket into the 'Post office' queue" (TA6)

To be executed when the transition "Order placed" (T5) applied.

- "Close ticket successfully" (TA7)

  To be executed when:

  - The transition "Shipment received" (T6) applied

- "Close ticket unsuccessfully" (TA8)

  To be executed when:

  - The Transition "Approval denied" (T2) applied
  - The Transition "Order denied" (T4) applied

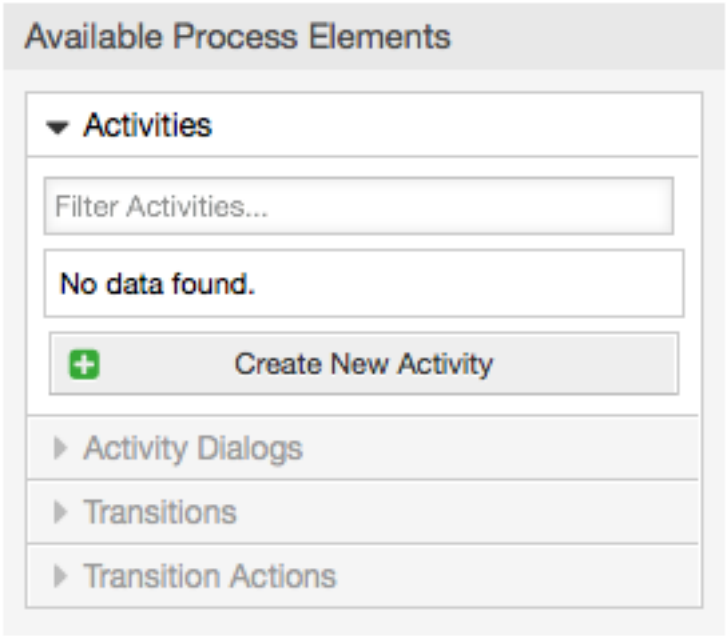
As you can see, there are places where the same Transition Actions should be executed. Therefore it is reasonable to make it possible to link Transition Actions freely with Transitions to be able to reuse them.

2.3.5. Activities

We chose the approach to see Activities as a basket which can contain one or more Activity Dialogs.

Click on "Activities" in the "Available Process Elements", then click on "Create New Activity".

Figure 5.17. Create New Activity button



In the opened popup screen fill the "Activity Name", then drag the required Activity Dialogs from the "Available Activity Dialogs" pool, and drop them into to the "Assigned Activity Dialogs" pool. This dialogs will be presented (in the ticket zoom screen) in the same order as it is defined on this screen translating from top to bottom, from left to right.

This order is specially important in the first Activity, since the first Activity Dialog for this activity is the only one that is presented when the process starts.

Create the following Activities:

- "Recording the demand" (A1)

  Contains the Activity Dialog "Recording the demand" (AD1)

- "Approval" (A2)

  Contains the Activity Dialogs "Approval denied" (AD2) as well as "Approved" (AD3)

- "Order" (A3)

  Contains the Activity Dialogs "Order rejected" (AD4) as well as "Order placed" (AD5)

- "Incoming" (A4)

  Contains the Activity Dialog "Shipment received" (AD6)

- "Process complete" (A5): This is an Activity without possible Activity Dialogs. It will be set after "Approval denied", "Order denied" or "Shipment received" and represents the end of the process.

Now we can clearly see that Activities are precisely defined states of a process ticket. After a successful Transition a process ticket moves from one Activity to another.

2.3.6. Book ordering process Path

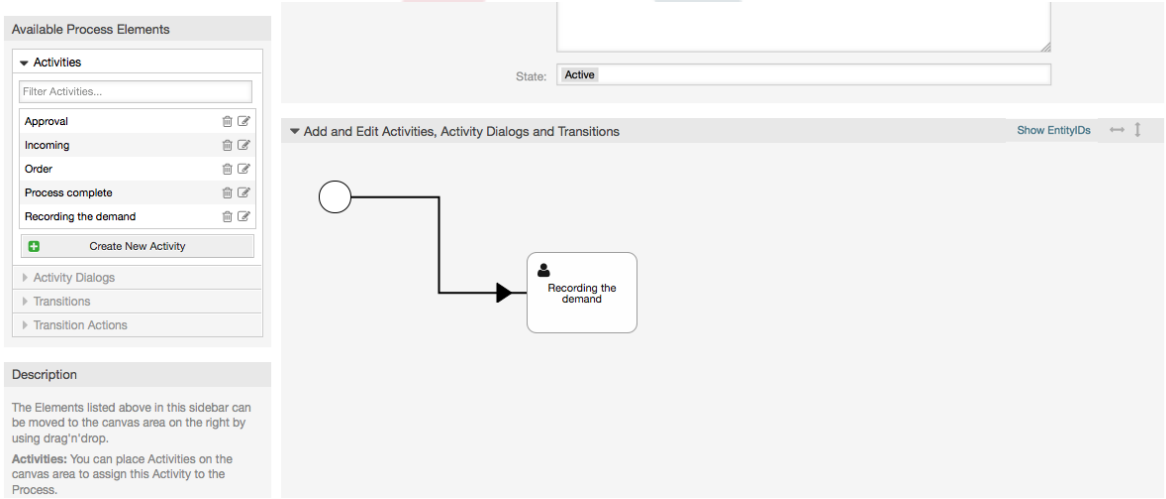
Let us conclude our example with the last missing piece in the puzzle, the Process as the a flow describer. In our case this is the whole ordering work-flow. Other processes could be office supply ordering or completely different processes.

The process has a starting point which consists of the start Activity and the start Activity Dialog. For any new book order, the start Activity Dialog (first Activity Dialog for the first Activity) is the first screen that is displayed. If this is completed and saved, the Process ticket will be created and can follow the configured work-flow.

The process also contains the directions for how the process ticket can move through the Process. Let's call this the "Path". It consists of the start Activity, one or more Transitions (possibly with Transition Actions), and other Activities.

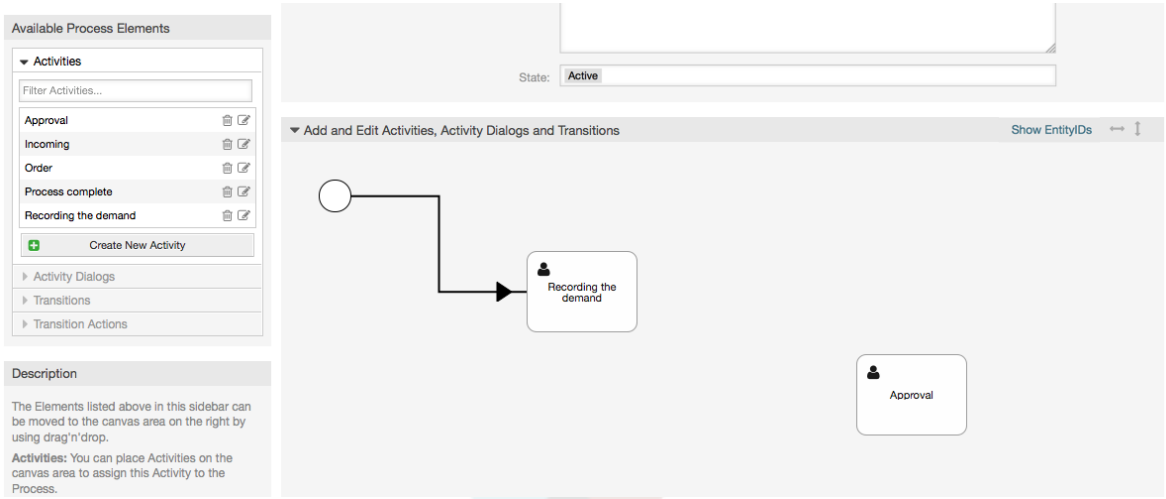
Assuming that the Activities has already assigned their Activity Dialogs drag an Activity from the accordion (in the left part of the screen) and drop it into the canvas area (below process information). Notice that an arrow from the process start (green circle) to the Activity is placed automatically. (This is the first Activity and its first Activity Dialog is the first screen that will be shown when the process starts).

Figure 5.18. Drag first Activity into the canvas



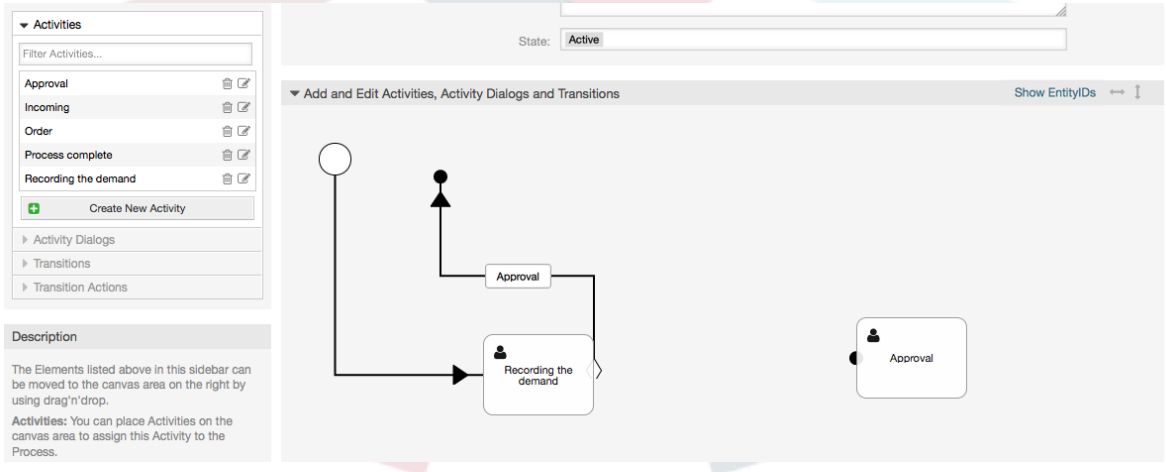
Next, drag another Activity into the canvas too. Now we will have two Activities in the canvas. The first one is connected to the start point and the second has no connections. You can hover the mouse over each activity to reveal their own Activity Dialogs.

Figure 5.19. Drag second Activity into the canvas



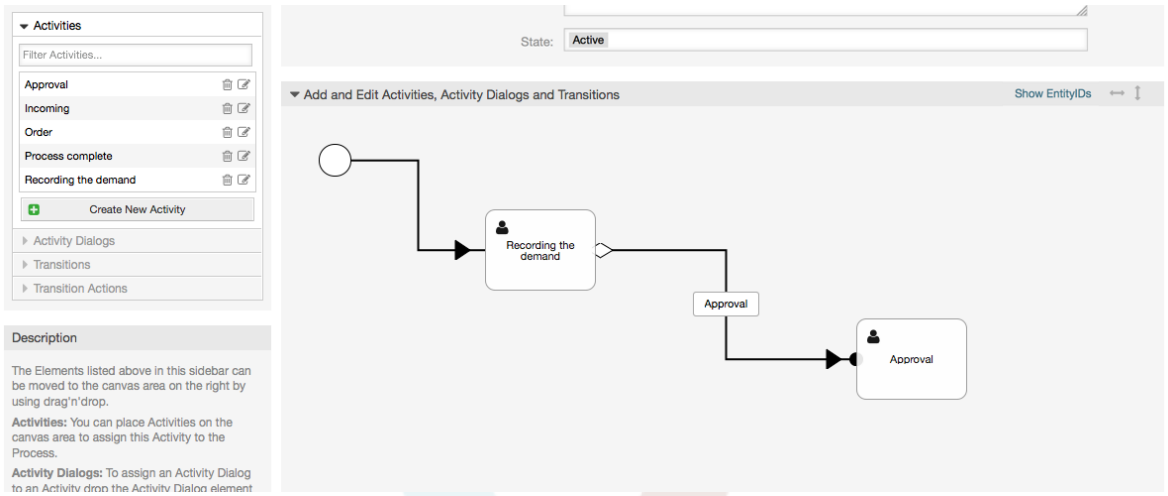
Then let's create the "Path" (connection) between this two Activities, for this we will use the Transitions. Click on Transitions in the accordion drag a Transition and drop it inside the first Activity. Notice that the Activity changes its color indicating that the Transition is attached. As soon as the Transition is dropped the end point of the Transition arrow will be placed next to the process start point. Drag the Transition arrow end point and drop it inside the other Activity to create the connection between the Activities.

Figure 5.20. Drag a Transition into the canvas



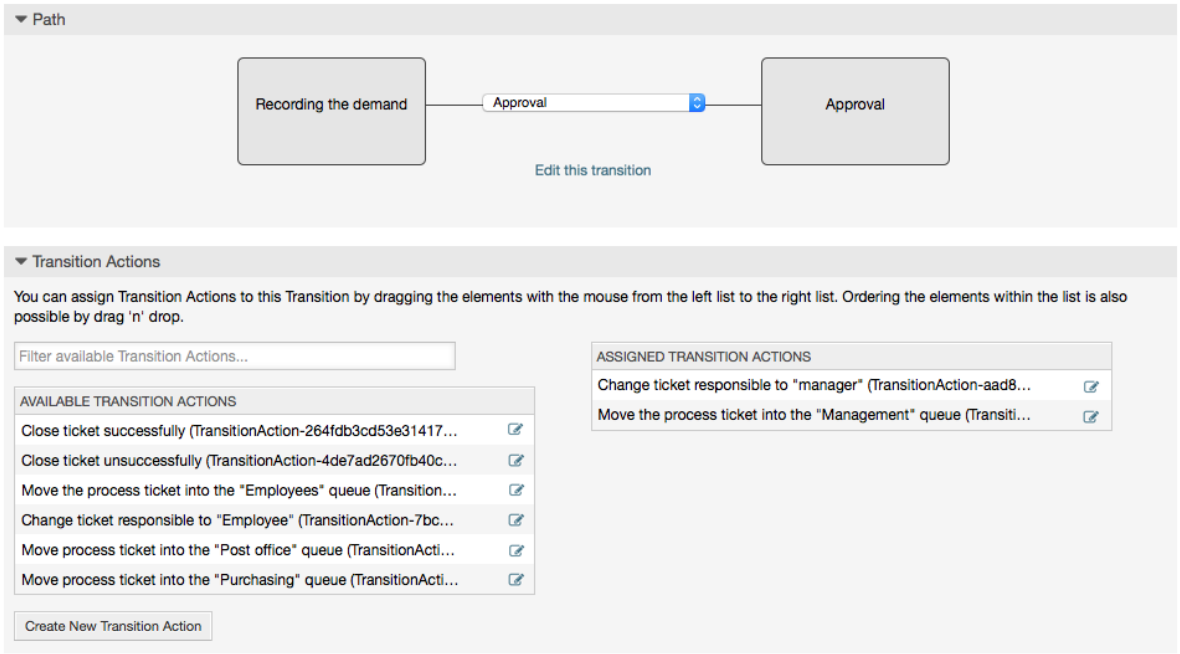
Now that the "Path" between the Actions is defined, then we need to assign the Transition Actions to the Transition, double click the Transition label (in the canvas), this will open a new popup window.

Figure 5.21. Connect Activities using Transitions



Drag the needed Transition Actions from Available Transition Actions pool and drop them into the Assigned Transition Actions pool and click on submit button.

Figure 5.22. Assign Transition Actions



Then back in the main process edit screen click on save button below the canvas to save all other changes.

Complete the "path" adding the following Activities, Transitions and Transition Actions:

Recording the demand until "Approval"

- Starting point: Activity: "Recording the demand" (A1)
- Possible Transition: "Approval" (T1)
  - If the condition of this activity is fulfilled, the ticket will move to Activity: "Approval" (A2)
  - Additionally, the following Transition Actions are executed:

- "Move the process ticket into the 'Management' queue" (TA1)
- "Change ticket responsible to 'manager'" (TA2)

The Activity: "Recording the demand" (A1) is a defined step of the process ticket, where there is the possibility for the Transition: "Approval" (T1). If this applies, the ticket will move to the next Activity: "Approval" (A2), and the Transition Actions: "Move the process ticket into the 'Management' queue" (TA1) and "Change ticket responsible to 'manager'" (TA2) are executed. In the Activity: "Approval" (A2), the Activity Dialogs: "Approval denied" (AD2) and "Approved" (AD3) are available.

Approval

- Starting Point: Activity "Approval" (A2)
- Possible Transitions:
  - "Approval denied" (T2)
    - If this matches, the process ticket will move to Activity: "Process complete" (A5).
    - Additionally, the following Transition Actions are executed:
      - "Move process ticket into the 'Employees' queue" (TA3)
      - "Change ticket responsible to 'Employee'" (TA4)
      - "Close ticket unsuccessfully" (TA8)
  - "Approved" (T3)
    - If this matches, the process ticket will move to Activity: "Order" (A3).
    - Additionally, the following Transition Action is executed:
      - "Move process ticket into the 'Purchasing' queue" (TA5)

We can see that from the current Activity, which defines a step of the process ticket, there are one or more possibilities for Transition which have exactly one target Activity (and possibly one or more Transition Actions).

Order

- Starting Point: Activity "Order" (A3)
- Possible Transitions:
  - "Order denied" (T4)
    - If this matches, the process ticket will move to Activity: "Process complete" (A5).
    - Additionally, the following Transition Actions are executed:
      - "Move process ticket into the 'Employees' queue" (TA3)
      - "Set ticket responsible to 'Employee'" (TA4)
      - "Close ticket unsuccessfully" (TA8)
  - "Order placed" (T5)
    - If this matches, the process ticket will move to Activity: "Incoming" (A4).

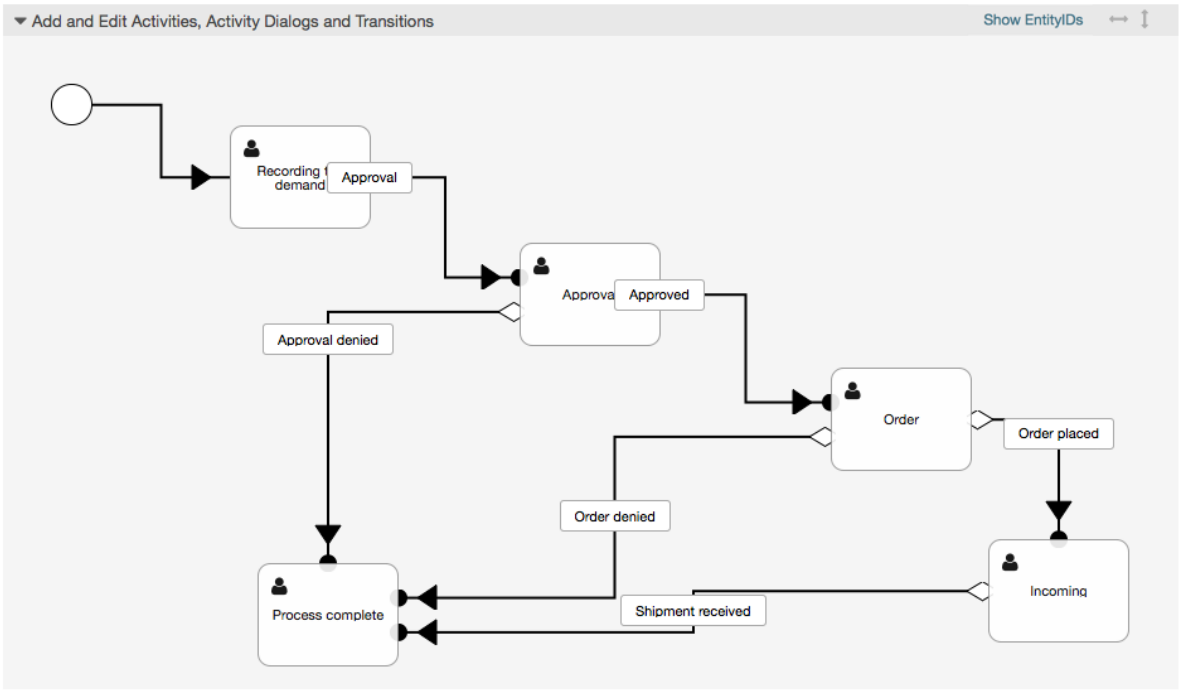
- Additionally, the following Transition Action is executed:
  - "Move process ticket into the 'Post office' queue" (TA6)

Incoming

- Starting Point: Activity "Incoming" (A4)
- Possible Transitions:
  - "Shipment received" (T6)
    - If this matches, the process ticket will move to Activity: "Process complete" (A5).
    - Additionally, the following Transition Actions are executed:
      - "Move process ticket into the 'Employees' queue" (TA3)
      - "Set ticket responsible to 'Employee'" (TA4)
      - "Close ticket successfully" (TA7)

The complete Path for the book ordering Process will then look like this:

Figure 5.23. Book ordering complete process path



After you finish the process path please click on "Save" button in the lower part of the canvas and then click on "Synchronize All Processes" button. This will gather all processes information form the Database and create a cache file (in Perl language). This cache file is actually the processes configuration that the system will use to create or use process tickets.

Any change that is made of the process (in the GUI) will require to re-synchronize the cache file in order to get the change reflected in the system.

It is also possible to import the whole process from a YAML file, but it is still necessary to create all Dynamic Fields, Users, Queues, etc that are needed by each process before the import.

Notice that if the process requires the use of ACLs those are also needed to be set manually.

The following is the complete YAML file for the book ordering process example:

```
---
Activities:
  A1:
    ActivityDialogs:
      - AD1
    ChangeTime: 2012-11-23 14:49:22
    Config:
      ActivityDialog:
        1: AD1
    CreateTime: 2012-11-23 11:49:38
    EntityID: A1
    ID: 151
    Name: Recording the demand
  A2:
    ActivityDialogs:
      - AD2
      - AD3
    ChangeTime: 2012-12-13 00:55:12
    Config:
      ActivityDialog:
        1: AD2
        2: AD3
    CreateTime: 2012-11-23 11:50:11
    EntityID: A2
    ID: 152
    Name: Approval
  A3:
    ActivityDialogs:
      - AD4
      - AD5
    ChangeTime: 2012-11-23 18:12:14
    Config:
      ActivityDialog:
        1: AD4
        2: AD5
    CreateTime: 2012-11-23 11:50:35
    EntityID: A3
    ID: 153
    Name: Order
  A4:
    ActivityDialogs:
      - AD6
    ChangeTime: 2012-11-23 18:12:35
    Config:
      ActivityDialog:
        1: AD6
    CreateTime: 2012-11-23 11:51:00
    EntityID: A4
    ID: 154
    Name: Incoming
  A5:
    ActivityDialogs: []
    ChangeTime: 2012-11-23 11:51:33
    Config: {}
    CreateTime: 2012-11-23 11:51:33
    EntityID: A5
    ID: 155
    Name: Process complete
ActivityDialogs:
  AD1:
    ChangeTime: 2012-12-06 02:16:21
    Config:
      DescriptionLong: ''
      DescriptionShort: Recoding the demand
      FieldOrder:
        - DynamicField_Author
```

```
      - DynamicField_ISBN
      - DynamicField_Title
      - DynamicField_Status
    Fields:
      DynamicField_Author:
        DefaultValue: ''
        DescriptionLong: ''
        DescriptionShort: ''
        Display: 1
      DynamicField_ISBN:
        DefaultValue: ''
        DescriptionLong: ''
        DescriptionShort: ''
        Display: 1
      DynamicField_Status:
        DefaultValue: ''
        DescriptionLong: ''
        DescriptionShort: ''
        Display: 1
      DynamicField_Title:
        DefaultValue: ''
        DescriptionLong: ''
        DescriptionShort: ''
        Display: 1
    Interface:
      - AgentInterface
    Permission: ''
    RequiredLock: 0
    SubmitAdviceText: ''
    SubmitButtonText: ''
  CreateTime: 2012-11-23 14:34:43
  EntityID: AD1
  ID: 154
  Name: Recording the demand
AD2:
  ChangeTime: 2012-11-23 14:57:41
  Config:
    DescriptionLong: ''
    DescriptionShort: Approval denied
    FieldOrder:
      - Article
      - DynamicField_Status
    Fields:
      Article:
        Config:
          ArticleType: note-internal
        DefaultValue: ''
        DescriptionLong: ''
        DescriptionShort: ''
        Display: 1
      DynamicField_Status:
        DefaultValue: ''
        DescriptionLong: ''
        DescriptionShort: ''
        Display: 1
    Interface:
      - AgentInterface
    Permission: ''
    RequiredLock: 0
    SubmitAdviceText: ''
    SubmitButtonText: Deny Request
  CreateTime: 2012-11-23 14:36:39
  EntityID: AD2
  ID: 155
  Name: Approval denied
AD3:
  ChangeTime: 2012-12-14 03:14:23
  Config:
    DescriptionLong: ''
    DescriptionShort: Approved
    FieldOrder:
      - DynamicField_Status
```

```
    Fields:
      DynamicField_Status:
        DefaultValue: ''
        DescriptionLong: ''
        DescriptionShort: ''
        Display: 1
    Interface:
      - AgentInterface
    Permission: ''
    RequiredLock: 0
    SubmitAdviceText: ''
    SubmitButtonText: Approve Request
  CreateTime: 2012-11-23 14:37:35
  EntityID: AD3
  ID: 156
  Name: Approved
AD4:
  ChangeTime: 2012-11-23 14:58:52
  Config:
    DescriptionLong: ''
    DescriptionShort: Order rejected
    FieldOrder:
      - Article
      - DynamicField_Status
    Fields:
      Article:
        Config:
          ArticleType: note-internal
        DefaultValue: ''
        DescriptionLong: ''
        DescriptionShort: ''
        Display: 1
      DynamicField_Status:
        DefaultValue: ''
        DescriptionLong: ''
        DescriptionShort: ''
        Display: 1
    Interface:
      - AgentInterface
    Permission: ''
    RequiredLock: 0
    SubmitAdviceText: ''
    SubmitButtonText: Reject Order
  CreateTime: 2012-11-23 14:38:48
  EntityID: AD4
  ID: 157
  Name: Order rejected
AD5:
  ChangeTime: 2012-12-06 02:20:12
  Config:
    DescriptionLong: ''
    DescriptionShort: Order placed
    FieldOrder:
      - DynamicField_DeliveryDate
      - DynamicField_Price
      - DynamicField_Supplier
      - DynamicField_Status
    Fields:
      DynamicField_DeliveryDate:
        DefaultValue: ''
        DescriptionLong: ''
        DescriptionShort: ''
        Display: 1
      DynamicField_Price:
        DefaultValue: ''
        DescriptionLong: ''
        DescriptionShort: ''
        Display: 1
      DynamicField_Status:
        DefaultValue: ''
        DescriptionLong: ''
        DescriptionShort: ''
```

```
          Display: 1
        DynamicField_Supplier:
          DefaultValue: ''
          DescriptionLong: ''
          DescriptionShort: ''
          Display: 1
      Interface:
        - AgentInterface
      Permission: ''
      RequiredLock: 0
      SubmitAdviceText: ''
      SubmitButtonText: Place Order
    CreateTime: 2012-11-23 14:41:28
    EntityID: AD5
    ID: 158
    Name: Order placed
  AD6:
    ChangeTime: 2012-11-23 14:42:43
    Config:
      DescriptionLong: ''
      DescriptionShort: Shipment received
      FieldOrder:
        - DynamicField_DateOfReceipt
        - DynamicField_Status
      Fields:
        DynamicField_DateOfReceipt:
          DefaultValue: ''
          DescriptionLong: ''
          DescriptionShort: ''
          Display: 1
        DynamicField_Status:
          DefaultValue: ''
          DescriptionLong: ''
          DescriptionShort: ''
          Display: 1
      Interface:
        - AgentInterface
      Permission: ''
      RequiredLock: 0
      SubmitAdviceText: ''
      SubmitButtonText: ''
    CreateTime: 2012-11-23 14:42:43
    EntityID: AD6
    ID: 159
    Name: Shipment received
Process:
  Activities:
    - A1
    - A2
    - A3
    - A4
    - A5
  ChangeTime: 2012-12-06 02:31:59
  Config:
    Description: The process to order a book
    Path:
      A1:
        T1:
          ActivityEntityID: A2
          TransitionAction:
            - TA2
            - TA1
      A2:
        T2:
          ActivityEntityID: A5
          TransitionAction:
            - TA3
            - TA4
            - TA8
        T3:
          ActivityEntityID: A3
          TransitionAction:
```

```
            - TA5
      A3:
        T4:
          ActivityEntityID: A5
          TransitionAction:
            - TA3
            - TA4
            - TA8
        T5:
          ActivityEntityID: A4
          TransitionAction:
            - TA6
      A4:
        T6:
          ActivityEntityID: A5
          TransitionAction:
            - TA3
            - TA4
            - TA7
      A5: {}
    StartActivity: A1
    StartActivityDialog: AD1
  CreateTime: 2012-11-23 11:45:12
  EntityID: P1
  ID: 94
  Layout:
    A1:
      left: 172
      top: 63
    A2:
      left: 402
      top: 156
    A3:
      left: 649
      top: 255
    A4:
      left: 774
      top: 391
    A5:
      left: 194
      top: 410
  Name: Book ordering
  State: Active
  StateEntityID: S1
  TransitionActions:
    - TA1
    - TA2
    - TA3
    - TA4
    - TA8
    - TA5
    - TA3
    - TA4
    - TA8
    - TA6
    - TA3
    - TA4
    - TA7
  Transitions:
    - T1
    - T2
    - T3
    - T4
    - T5
    - T6
TransitionActions:
  TA1:
    ChangeTime: 2012-11-23 16:01:37
    Config:
      Config:
        Queue: Management
      Module: Kernel::System::ProcessManagement::TransitionAction::TicketQueueSet
```

```
  CreateTime: 2012-11-23 15:50:59
  EntityID: TA1
  ID: 61
  Name: Move the process ticket into the "Management" queue
TA2:
  ChangeTime: 2012-11-23 16:02:12
  Config:
    Config:
      Responsible: manager
    Module: Kernel::System::ProcessManagement::TransitionAction::TicketResponsibleSet
  CreateTime: 2012-11-23 15:58:22
  EntityID: TA2
  ID: 62
  Name: Change ticket responsible to "manager"
TA3:
  ChangeTime: 2012-11-24 14:27:02
  Config:
    Config:
      Queue: Employees
    Module: Kernel::System::ProcessManagement::TransitionAction::TicketQueueSet
  CreateTime: 2012-11-23 16:02:54
  EntityID: TA3
  ID: 63
  Name: Move the process ticket into the "Employees" queue
TA4:
  ChangeTime: 2012-11-23 16:04:06
  Config:
    Config:
      Responsible: Employee
    Module: Kernel::System::ProcessManagement::TransitionAction::TicketResponsibleSet
  CreateTime: 2012-11-23 16:04:06
  EntityID: TA4
  ID: 64
  Name: Change ticket responsible to "Employee"
TA5:
  ChangeTime: 2012-12-06 02:18:34
  Config:
    Config:
      Queue: Purchasing
    Module: Kernel::System::ProcessManagement::TransitionAction::TicketQueueSet
  CreateTime: 2012-11-23 16:04:54
  EntityID: TA5
  ID: 65
  Name: Move process ticket into the "Purchasing" queue
TA6:
  ChangeTime: 2012-12-06 02:18:48
  Config:
    Config:
      Queue: Post office
    Module: Kernel::System::ProcessManagement::TransitionAction::TicketQueueSet
  CreateTime: 2012-11-23 16:06:20
  EntityID: TA6
  ID: 66
  Name: Move process ticket into the "Post office" queue
TA7:
  ChangeTime: 2012-12-06 02:29:55
  Config:
    Config:
      State: closed successful
    Module: Kernel::System::ProcessManagement::TransitionAction::TicketStateSet
  CreateTime: 2012-12-06 02:29:27
  EntityID: TA7
  ID: 67
  Name: Close ticket successfully
TA8:
  ChangeTime: 2012-12-06 02:31:12
  Config:
    Config:
      State: closed unsuccessful
    Module: Kernel::System::ProcessManagement::TransitionAction::TicketStateSet
  CreateTime: 2012-12-06 02:31:12
  EntityID: TA8
```

```
    ID: 68
    Name: Close ticket unsuccessfully
Transitions:
  T1:
    ChangeTime: 2012-11-23 15:12:20
    Config:
      Condition:
        1:
          Fields:
            DynamicField_Status:
              Match: Approval
              Type: String
          Type: and
      ConditionLinking: and
    CreateTime: 2012-11-23 11:53:52
    EntityID: T1
    ID: 94
    Name: Approval
  T2:
    ChangeTime: 2012-11-23 15:12:50
    Config:
      Condition:
        1:
          Fields:
            DynamicField_Status:
              Match: Approval denied
              Type: String
          Type: and
      ConditionLinking: and
    CreateTime: 2012-11-23 11:54:26
    EntityID: T2
    ID: 95
    Name: Approval denied
  T3:
    ChangeTime: 2012-11-23 15:13:29
    Config:
      Condition:
        1:
          Fields:
            DynamicField_Status:
              Match: Approved
              Type: String
          Type: and
      ConditionLinking: and
    CreateTime: 2012-11-23 11:54:54
    EntityID: T3
    ID: 96
    Name: Approved
  T4:
    ChangeTime: 2012-11-23 15:14:08
    Config:
      Condition:
        1:
          Fields:
            DynamicField_Status:
              Match: Order denied
              Type: String
          Type: and
      ConditionLinking: and
    CreateTime: 2012-11-23 11:55:25
    EntityID: T4
    ID: 97
    Name: Order denied
  T5:
    ChangeTime: 2012-11-23 18:30:33
    Config:
      Condition:
        1:
          Fields:
            DynamicField_Status:
              Match: Order placed
              Type: String
```

```
        Type: and
      ConditionLinking: and
    CreateTime: 2012-11-23 11:56:15
    EntityID: T5
    ID: 98
    Name: Order placed
  T6:
    ChangeTime: 2012-11-23 15:15:30
    Config:
      Condition:
        1:
          Fields:
            DynamicField_Status:
              Match: Shipment received
              Type: String
          Type: and
      ConditionLinking: and
    CreateTime: 2012-11-23 11:56:48
    EntityID: T6
    ID: 99
    Name: Shipment received
```

2.4. Process configuration reference

2.4.1. Process

A Process models the path of a workflow/process. The waypoints on this path can be Activities or Transitions, we'll talk about these later.

2.4.1.1. Process configuration

The Process configuration can be done in the file `Kernel/Config.pm` but it is strongly recommended to create new files like `Kernel/Config/Files/MyProcess.pm`. Notice that the GUI generates the file `Kernel/Config/File/ZZZProcessManagement` please avoid to use that filename, otherwise it will be overwritten when you sync processes. Let's see an example process configuration (from process cache file):

```
$Self->{'Process'} = {
    'P1' => {
        Name                => 'Book order',
        CreateTime          => '16-02-2012 13:37:00',
        CreateBy            => '1',
        ChangeTime          => '17-02-2012 13:37:00',
        ChangeBy            => '1',
        State               => 'Active',
        StartActivity       => 'A1',
        StartActivityDialog => 'AD1',
        Path => {
            'A1' => {
                'T1' => {
                    ActivityEntityID => 'A2',
                },
            },
            'A2' => {
                'T2' =>  {
                    ActivityEntityID => 'A3',
                },
            },
        },
    },
    'P2' => {
        Name                => 'IT order',
        CreateTime          => '26-02-2012 13:37:00',
        CreateBy            => '1',
        ChangeTime          => '27-02-2012 13:37:00',
```

```
        ChangeBy            => '1',
        State               => 'Active',
        StartActivity       => 'A2',
        StartActivityDialog => 'AD2',
        Path => {
            'A2' => {
                'T3' => {
                    ActivityEntityID => 'A4',
                },
            },
        },
    }
};
```

2.4.1.2. Name

The name of the process, this can be selected by the agent when creating a new process ticket.

2.4.1.3. CreateTime

The time when the process was created.

2.4.1.4. CreateBy

The UID of the user creating the process.

2.4.1.5. ChangeTime

The time when the process was changed.

2.4.1.6. ChangeBy

The UID of the user who made the last change to the process.

2.4.1.7. State

Defines the state of a process. Possible values:

- 'Active' are all processes which can be used in new process tickets.
- 'FadeAway' are processes which cannot be selected any more for new tickets, but existing tickets still can use the process.
- 'Inactive' processes are deactivated and cannot be used for new or existing tickets.

2.4.1.8. StartActivity

When creating a new process ticket, a StartActivity must be defined. As soon as the ticket is created, this Activity will be set and used as the base for the first transition checks.

2.4.1.9. StartActivityDialog

For new process tickets, a StartActivityDialog must be defined. This will be shown when creating a new process ticket (after the process was selected). At this point, the ticket does not exist yet, it will be created after submitting the StartActivityDialog.

2.4.1.10. Path

The Path contains the structure of the Activities, and the possible Transitions between them, for the current process. And also the Transition Actions that happens when transitioning . This controls the way that a process ticket can take. Example:

```
'A1' => {
    'T1' => {
        ActivityEntityID => 'A2',
    },
    'T2' => {
        ActivityEntityID => 'A3',
    },
    'T3' => {
        ActivityEntityID => 'A4',
        TransitionAction => ['TA1', 'TA2'],
    },
},
```

If a process ticket is in Activity 'A1', it has three possible ways to get to another Activity. In the Transitions 'T1' to 'T3', conditions are defined, that a process ticket must fulfill to move (transit) to another Activity.

If in this case all the values of the process ticket and its dynamic fields that are needed for the Transition 'T2' are correct, the ticket will be moved from Activity 'A1' to 'A3'. After an ActivityDialog is submitted, or any other change is made to a ticket, it will be checked for possible Transitions from the current Activity. If multiple Transitions are possible, the first one will be used (based on numerical sorting of the TransitionIDs).

Additionally, it is possible to assign Transition Actions to Transitions in the Path configuration. These are modules which are executed after a successful Transition. They have to be specified in array form as in the example, we'll talk about the details later.

2.4.2. Activity

An Activity contains one or more Activity Dialogs and models a 'step' in the process. All Activity Dialogs of the current Activity are displayed in the ticket zoom and can be used until the conditions of a Transition are fulfilled.

2.4.2.1. Activity configuration

Let's see an example activity configuration:

```
$Self->{'Process::Activity'} =
{
    'A1' => {
        Name       => 'Activity 1 optional',
        CreateTime => '16-02-2012 13:37:00',
        CreateBy   => '1',
        ChangeTime => '17-02-2012 13:37:00',
        ChangeBy   => '1',
        ActivityDialog => {
            1 => 'AD1',
        },
    },
    'A2' => {
        Name       => 'Activity 2 optional',
        CreateTime => '16-02-2012 13:37:00',
        CreateBy   => '1',
        ChangeTime => '17-02-2012 13:37:00',
        ChangeBy   => '1',
        ActivityDialog => {
            1 => 'AD5',
            2 => 'AD6',
            3 => 'AD1',
        },
    },
};
```

2.4.2.2. Name

The name of the activity.

2.4.2.3. CreateTime

The time when it was created.

2.4.2.4. CreateBy

UID of the user who created the Activity.

2.4.2.5. ChangeTime

The last time when it was changed.

2.4.2.6. ChangeBy

UID of the last user who changed the Activity.

2.4.2.7. ActivityDialog

Activity Dialog contains the list of Activity Dialogs which are available in this Activity. All Activity Dialogs of the current Activity are displayed in the ticket zoom. Their order is set by the order in the configuration, here 'AD5' is shown before 'AD6' and 'AD1'.

2.4.3. ActivityDialog

An Activity Dialog is a particular screen and can be used in different Activities.

2.4.3.1. ActivityDialog configuration

Let's see an example config:

```
$Self->{'Process::ActivityDialog'} = {
    'AD1' => {
        Name             => 'ActivityDialog 1 optional',
        DescriptionShort => 'Basic info',
        DescriptionLong  => 'Please insert the necessesary basic information for IT orders',
        CreateTime       => '28-02-2012 13:37:00',
        CreateBy         => '1',
        ChangeTime       => '29-02-2012 13:37:00',
        ChangeBy         => '1',
        Fields => {
            PriorityID => {
                DescriptionShort => 'Priority ID',
                DescriptionLong  => 'Enter the priority here',
                Display          => 2,
            },
        },
        FieldOrder       => [ 'PriorityID' ],
        SubmitAdviceText => 'Note: If you submit the form...',
        SubmitButtonText => 'Send request',
    },
    'AD2' => {
        Name             => 'ActivityDialog 2 optional',
        DescriptionShort => 'Basic info',
        DescriptionLong  => 'Please insert the necessesary basic information for Book
 orders',
        CreateTime       => '28-02-2012 13:37:00',
        CreateBy         => '1',
        ChangeTime       => '29-02-2012 13:37:00',
        ChangeBy         => '1',
        Fields => {
```

```
            StateID => {
                DescriptionShort => 'State ID',
                DescriptionLong  => 'Enter the state here',
                Display          => 2,
                DefaultValue     => '2',
            },
            Queue => {
                DescriptionShort => 'Queue ID',
                DescriptionLong  => 'Enter the queue here',
                Display          => 2,
                DefaultValue     => 'Raw',
            },
            Title => {
                DescriptionShort => 'Title',
                DescriptionLong  => 'Enter the title here',
                Display          => 1,
                DefaultValue     => 'Default Title',
            },
            DynamicField_Anzahl => {
                DescriptionShort => 'Amount',
                DescriptionLong  => 'Enter the amount here',
                Display          => 2,
                DefaultValue     => '4',
            },
        },
        FieldOrder       => [ 'DynamicField_Anzahl', 'StateID', 'Queue', 'Title' ],
        SubmitAdviceText => 'Note: If you submit the form...',
        SubmitButtonText => 'Send request',
    },
};
```

2.4.3.2. Name

Name of the Activity Dialog.

2.4.3.3. CreateTime

The time when it was created.

2.4.3.4. CreateBy

UID of the user who created this Activity Dialog.

2.4.3.5. ChangeTime

The last time when it was changed.

2.4.3.6. ChangeBy

UID of the last user who changed this Activity Dialog.

2.4.3.7. Fields

Contains all fields which can be displayed in this Activity Dialog. The following fields can currently be used:

```
    Title
    State
    StateID
    Priority
    PriorityID
    Lock
    LockID
    Queue
    QueueID
```

```
    Customer
    CustomerID
    CustomerNo
    CustomerUserID
    Owner
    OwnerID
    Type
    TypeID
    SLA
    SLAID
    Service
    ServiceID
    Responsible
    ResponsibleID
    PendingTime
    DynamicField_$FieldName  # for all dynamic fields
```

Example of a single field configuration:

```
StateID => {
    DescriptionShort => 'State ID',
    DescriptionLong  => 'Enter the state here',
    Display          => 2,
    DefaultValue     => '2',
},
```

The field "Article" is a special case. If it is present in a "Fields" configuration, the Activity Dialog will contain a complete Richtext editor with subject field and attachment handling. The entered text will then be added to the ticket as an article and sent by email. Let's see an example Article field configuration:

```
Article => {
    DescriptionShort => 'Please insert your comment here.',
    DescriptionLong => '',
    Display         => 1,
    Config          => {
        ArticleType  => 'note-internal',
        LabelSubject => '',
        LabelBody    => '',
    },
},
```

Let's look at the field configuration options:

2.4.3.7.1. DescriptionShort

Optional short description that is shown with the field title.

2.4.3.7.2. DescriptionLong

Optional longer field description that is shown then the mouse is over the field, for example advice on how to fill out the field.

2.4.3.7.3. Display

Controls if the field is shown and/or mandatory. Possible values:

- '0': field is invisible. This can be helpful if field values should automatically be set. The configured DefaultValue will be stored in this case.
- '1': field is visible, but optional.
- '2': field is visible and mandatory. The following fields can only be invisible or mandatory:

```
    QueueID
    Queue
    State
    StateID
    Lock
    LockID
    Priority
    PriorityID
    Type
    TypeID
```

If fields are configured as optional, and no value is submitted by the user, the Default Value will be saved when the Activity Dialog is submitted by the user.

2.4.3.7.4. DefaultValue

For fields with 'ID' (like QueueID, OwnerID), this refers to the database ID of the value. For other fields without 'ID' (like Queue, Owner), the DefaultValue must contain the value itself. Example:

```
Queue => {
    DescriptionShort => 'Queue',
    DescriptionLong  => 'Enter the queue here',
    Display          => 2,
    DefaultValue     => 'Raw',
},
```

2.4.3.8. FieldOrder

Here the display order of the fields is configured. IMPORTANT: Invisible fields also must be configured here, because only configured fields will be considered when saving. Fields which are not configured will not be saved.

2.4.3.9. SubmitAdviceText

Optional text to be shown right above the submit button for additional help or advice text.

2.4.3.10. SubmitButtonText

Optional custom text for the submit button.

2.4.4. Transition

A Transition decides - based on configurable conditions - which path in the Process is taken, i. e. to which Activity a Process ticket can be moved.

2.4.4.1. Transition configuration

Let's see an example:

```
$Self->{'Process::Transition'} = {
    'T1' => {
        Name => 'Transition 1',
        CreateTime => '14-03-2012 13:37:00', # optional
        CreateBy   => '1',                   # optional
        ChangeTime => '15-03-2012 13:37:00', # optional
        ChangeBy   => '15-03-2012 13:37:00', # optional
        Condition  => {
            Cond1  => {
                Fields => {
```

```
                    StateID => {
                        Type  => 'String',
                        Match => '1',
                    },
                },
            },
        },
    },
    'T2' => {
        Name       => 'Transition 2 optional',
        CreateTime => 'DATE',   # optional
        CreateBy   => 'USERID', # optional
        ChangeTime => 'DATE',   # optional
        ChangeBy => 'USERID',   # optional
        Condition => {
            Cond1 => {
                Queue               => 'Raw',
                DynamicField_Farbe  => '2',
                DynamicField_Anzahl => '1',
            },
        },
    },
};
```

2.4.4.2. Name

Name of the transition.

2.4.4.3. CreateTime

Time when it was created.

2.4.4.4. CreateBy

UID of the user who created this Transition.

2.4.4.5. ChangeTime

Last time when it was changed.

2.4.4.6. ChangeBy

UID of the last user who changed this Transition.

2.4.4.7. Condition

Contains all conditions that are necessary for this Transition to take effect. Example:

```
Condition => {
    Type  => 'and',
    Cond1 => {
        Type   => 'and',
        Fields => {
            StateID => {
                Type  => 'String',
                Match => '1',
            },
            DynamicField_Marke => {
                Type  => 'String',
                Match => 'VW',
        },
    },
    Cond2 => {
        Type => 'and',
        Fields => {
            Queue => {
```

```
                Type  => 'String',
                Match => 'Raw',
            },
        },
    },
},
```

Let's look at the condition configuration in detail.

2.4.4.7.1. Type (Condition)

Specifies the way the different condition elements are connected to each other. Possible values:

- 'and': This is the default. All conditions must be met for the transition to take effect.
- 'or': At least one condition must match.
- 'xor': Exactly one condition must match, not more.

2.4.4.7.2. Cond1

This is the name of an example condition. It can be freely chosen. Conditions are evaluated in sorted order.

2.4.4.7.3. Type (Cond)

Specifies the way how the individual field tests of this condition are connected to each other. Possible values:

- 'and': This is the default. All field tests must match for this condition to match.
- 'or': At least one field test must match.
- 'xor': Exactly one field test must match, not more.

2.4.4.7.4. Fields

Specifies the particular fields whose values should be tested. From our example:

```
Fields => {
    StateID => {
        Type  => 'String',
        Match => '1',
    },
```

2.4.4.7.5. StateID

Example of a field name. The following ticket fields can be used:

```
    Title
    State
    StateID
    Priority
    PriorityID
    Lock
    LockID
    Queue
    QueueID
    Customer
    CustomerID
    CustomerNo
    CustomerUserID
```

```
    Owner
    OwnerID
    Type
    TypeID
    SLA
    SLAID
    Service
    ServiceID
    Responsible
    ResponsibleID
    DynamicField_$FieldName # for all DynamicFields
```

When testing a field with 'ID' (like SLAID), the database ID of the field will be used for testing, for other fields (like SLA) the actual value is used for testing.

2.4.4.7.6. Type

Determines the kind of field testing. Possible values:

- 'String': Compares the field value with the string specified in 'Match'. Matches if they are exactly the same.
- 'Hash': Compares the field value (hash) with the hash specified in 'Match'. All hash values must be the same.
- 'Array': Compares the field value (array) with the array specified in 'Match'. Both lists must be the same.
- 'Regex': The field value can be tested with a regular expression. It is important that 'Match' contains *qr{}xms* as a base condition. Between the braces the actual regular expression can be noted.
- 'Module': Allows you to use a perl module for condition checking. If it returns 1, the check was positive. You can find an example module in `Kernel/System/ProcessManagement/TransitionValidation/ValidateDemo.pm`.

2.4.5. Transition Actions

Transition Actions are actions which can be triggered after successfully applied transitions (when a process ticket moves from one activity to another). These Transition Actions can be used to perform different changes on the ticket, e. g. change the Queue or the Owner of the ticket, and you can also create your own Transition Actions to perform other complex changes.

2.4.5.1. Transition Action configuration

Let's see an example:

```
$Self->{'Process::TransitionAction'} = {
    'TA1' => {
        Name   => 'Queue Move',
        Module => 'Kernel::System::ProcessManagement::TransitionAction::TicketQueueSet',
        Config => {
            Queue  => 'Junk',
            UserID => 123,
        },
    },
};
```

2.4.5.2. Name

The name of the Transition Action.

2.4.5.3. Module

Specifies the Perl module to be used.

2.4.5.4. Config

This parameter contains all settings which are required for the module. Its content depends on the particular Transition Action module which is used. Please see the documentation of the individual modules for details. In our example, only the Queue must be specified. Nevertheless we are also sending UserID parameter, by using the UserID parameter. The transition action will be executed impersonating the user with the given UserID.

The use of UserID inside the "Config" parameter of a Transition Action is accepted by all Transition Actions (since OTRS 3.2.4). In this example it could be particularly important if the user that triggers the Transition does not have permissions to move the ticket to the queue 'Junk', while the user with the UserID 123 might have.

2.4.5.5. Reusing Transition Action modules

To use Transition Action modules multiple times, just specify several Transition Actions in your configuration. Example:

```
    $Self->{'Process::TransitionAction'} = {
        'TA1' => {
            Name   => 'Queue Move Junk',
            Module => 'Kernel::System::ProcessManagement::TransitionAction::TicketQueueSet',
            Config => {
                Queue => 'Junk',
            },
        },
        'TA2' => {
            Name   => 'Queue Move Raw',
            Module => 'Kernel::System::ProcessManagement::TransitionAction::TicketQueueSet',
            Config => {
                Queue => 'Raw',
            },
        },
    };
```

Here the same module is used to move a process ticket into the 'Raw' queue, and another time to move it into the junk queue. The Transition Action which must be used for a particular Transition is determined from the 'Path' setting of the Process configuration.

2.4.5.6. Available Transition Actions

OTRS comes with several Transition Actions that can be used in your processes. Here you can find their documentation and how they need to be configured.

2.4.5.6.1. DynamicFieldSet

Sets one or more dynamic fields at a process ticket. Example:

```
$Self->{'Process::TransitionAction'} = {
    'TA1' => {
        Name   => 'Set DynamicField MasterSlave to Master and Approved to 1',
        Module => 'Kernel::System::ProcessManagement::TransitionAction::DynamicFieldSet',
        Config => {
            MasterSlave => 'Master',
            Approved    => '1',

        },
    },
};
```

'Name' specifies the name of the configured TransitionAction.

'MasterSlave' and 'Approved' are given as examples of DynamicField names. The values of the fields ('Master' and '1') will be set by this TransitionAction.

2.4.5.6.2. TicketArticleCreate

Creates an article and can be used to create notes or email replies. Example:

```
$Self->{'Process::TransitionAction'} = {
    'TA1' => {
        Name   => 'Article Create Note Internal',
        Module =>
 'Kernel::System::ProcessManagement::TransitionAction::TicketArticleCreate',
        Config => {
            ArticleType      => 'note-internal',                                #
 note-external|phone|fax|sms|...
                                                                               #
 excluding any email type
            SenderType       => 'agent',                                       #
 agent|system|customer
            ContentType      => 'text/plain; charset=ISO-8859-15',             # or
 optional Charset & MimeType (e.g. 'text/html; charset=UTF-8')
            Subject          => 'some short description',                      #
 required
            Body             => 'the message text',                            #
 required
            HistoryType      => 'OwnerUpdate',                                 #
 EmailCustomer|Move|AddNote|PriorityUpdate|WebRequestCustomer|...
            HistoryComment   => 'Some free text!',
            From             => 'Some Agent <email@example.com>',              #
 not required but useful
            To               => 'Some Customer A <customer-a@example.com>',    #
 not required but useful
            Cc               => 'Some Customer B <customer-b@example.com>',    #
 not required but useful
            ReplyTo          => 'Some Customer B <customer-b@example.com>',    #
 not required
            InReplyTo        => '<asdasdasd.12@example.com>',                  #
 not required but useful
            References       => '<asdasdasd.1@example.com> <asdasdasd.12@example.com>', #
 not required but useful
            NoAgentNotify    => 0,                                             # if
 you don't want to send agent notifications
            AutoResponseType => 'auto reply',                                  #
 auto reject|auto follow up|auto reply/new ticket|auto remove

            ForceNotificationToUserID   => [ 1, 43, 56 ],                      # if
 you want to force somebody
            ExcludeNotificationToUserID => [ 43, 56 ],
                 # if you want full exclude somebody from notifications,
                 # will also be removed in To: line of article,
                 # higher prio as ForceNotificationToUserID
            ExcludeMuteNotificationToUserID => [ 43, 56 ],
                 # the same as ExcludeNotificationToUserID but only the
                 # sending gets muted, agent will still shown in To:
                 # line of article
        },
    },
};
```

'Name' specifies the name of the configured TransitionAction. It can be freely chosen, but should reflect the purpose of the configured action.

'ArticleType' defines the type of the article to be created. Possible values: phone, fax, sms, webrequest, note-internal, note-external and note-report.

'SenderType' defines the sender type of the article. Possible values: agent, system, customer.

'ContentType' defines the content type of the article. Possible values: 'text/plain; charset=ISO-8859-15' or any other valid charset and mime type.

'Subject' defines the article title. Mandatory.

'Body' defines the article content. Mandatory.

'HistoryType' defines the type of the history entry. Possible values: AddNote, ArchiveFlagUpdate, Bounce, CustomerUpdate, EmailAgent, EmailCustomer, EscalationResponseTimeNotifyBefore, EscalationResponseTimeStart, EscalationResponseTimeStop, EscalationSolutionTimeNotifyBefore, EscalationSolutionTimeStart, EscalationSolutionTimeStop, EscalationUpdateTimeNotifyBefore, EscalationUpdateTimeStart, EscalationUpdateTimeStop, FollowUp, Forward, Lock, LoopProtection, Merged, Misc, Move, NewTicket, OwnerUpdate, PhoneCallAgent, PhoneCallCustomer, PriorityUpdate, Remove, ResponsibleUpdate, SendAgentNotification, SendAnswer, SendAutoFollowUp, SendAutoReject, SendAutoReply, SendCustomerNotification, ServiceUpdate, SetPendingTime, SLAUpdate, StateUpdate, Subscribe, SystemRequest, TicketDynamicFieldUpdate, TicketLinkAdd, TicketLinkDelete, TimeAccounting, TypeUpdate, Unlock, Unsubscribe, WebRequestCustomer.

'HistoryComment' defines the content of the history entry.

'From', 'To', 'Cc' and 'ReplyTo' take email addresses in the notation specified above.

'InReplyTo' and 'References' take email message IDs.

'NoAgentNotify' - if set to 1, the email notification of the Agent will not be sent.

'AutoResponseType' can take the following values: auto follow up, auto reject, auto remove, auto reply, auto reply/new ticket.

'ForceNotificationToUserID', 'ExcludeNotificationToUserID', 'ExcludeMuteNotificationToUserID' can take a list of UserIDs that are either always notified, not notified or listed as notified but not actually sent a notification email.

2.4.5.6.3. TicketCreate

Creates a ticket with an article, the new ticket can be linked with process ticket. Example:

```
$Self->{'Process::TransitionAction'} = {
    'TA1' => {
        Name   => 'Ticket Create',
        Module => 'Kernel::System::ProcessManagement::TransitionAction::TicketCreate',
        Config => {

            # ticket required:
            Title         => 'Some Ticket Title',
            Queue         => 'Raw',              # or QueueID => 123,
            Lock          => 'unlock',
            Priority      => '3 normal',         # or PriorityID => 2,
            State         => 'new',              # or StateID => 5,
            CustomerID    => '123465',
            CustomerUser  => 'customer@example.com',
            OwnerID       => 'someuserlogin',    # or OwnerID => 123,

            # ticket optional:
            TN              => $TicketObject->TicketCreateNumber(), # optional
            Type            => 'Incident',           # or TypeID => 1, not required
            Service         => 'Service A',          # or ServiceID => 1, not required
            SLA             => 'SLA A',              # or SLAID => 1, not required
            ResponsibleID   => 123,                  # not required
            ArchiveFlag     => 'y',                  # (y|n) not required
```

```
            PendingTime     => '2011-12-23 23:05:00', # optional (for pending states)
            PendingTimeDiff => 123 ,                  # optional (for pending states)

            # article required:
            ArticleType      => 'note-internal',                        # note-external|
phone|fax|sms|...
                                                                        #   excluding any
 email type
            SenderType       => 'agent',                                # agent|system|
customer
            ContentType      => 'text/plain; charset=ISO-8859-15',      # or optional
 Charset & MimeType (e.g. 'text/html; charset=UTF-8')
            Subject          => 'some short description',               # required
            Body             => 'the message text',                     # required
            HistoryType      => 'OwnerUpdate',                          #
            EmailCustomer|Move|AddNote|PriorityUpdate|WebRequestCustomer|...
            HistoryComment   => 'Some free text!',

            # article optional:
            From             => 'Some Agent <email@example.com>',       # not required but
 useful
            To               => 'Some Customer A <customer-a@example.com>', # not required
 but useful
            Cc               => 'Some Customer B <customer-b@example.com>', # not required
 but useful
            ReplyTo          => 'Some Customer B <customer-b@example.com>', # not required
            MessageID        => '<asdasdasd.123@example.com>',          # not required but
 useful
            InReplyTo        => '<asdasdasd.12@example.com>',           # not required but
 useful
            References       => '<asdasdasd.1@example.com> <asdasdasd.12@example.com>', #
 not required but useful
            NoAgentNotify    => 0,                                      # if you don't want
 to send agent notifications
            AutoResponseType => 'auto reply'                            # auto reject|auto
 follow up|auto reply/new ticket|auto remove

            ForceNotificationToUserID   => [ 1, 43, 56 ],               # if you want to
 force somebody
            ExcludeNotificationToUserID => [ 43,56 ],                   # if you want full
 exclude somebody from notifications,
                                                                        # will also be
 removed in To: line of article,
                                                                        # higher prio as
 ForceNotificationToUserID
            ExcludeMuteNotificationToUserID => [ 43,56 ],               # the same as
 ExcludeNotificationToUserID but only the
                                                                        # sending gets
 muted, agent will still shown in To:
                                                                        # line of article
            TimeUnit                        => 123

            # other:
            DynamicField_NameX => $Value,
            LinkAs => $LinkType,                                        # Normal, Parent,
 Child, etc. (respective original ticket)
            UserID => 123,                                              # optional, to
 override the UserID from the logged user
        },
    },
};
```

'Name' specifies the name of the configured TransitionAction. It can be freely chosen, but should reflect the purpose of the configured action.

'Title' The ticket title.

'Queue' or 'QueueID' specifies the name or id of the queue to be used in the new ticket.

'Lock' or 'LockID' sets the lock status of the ticket.

'Priority' or 'PriorityID' specifies the name or id of the priority to be used in the new ticket.

'State' or 'StateID' specifies the name or id of the state to be used in the new ticket.

'CustomerID', the customer id to be set for the new ticket.

'CustomerUser', the login of the customer that will be assigned in the ticket.

'Owner' or 'OwnerID', specifies the login or id of the agent that will be the new ticket owner.

'TN', custom number for the new ticket.

'Type' or 'TypeID' specifies the name or id of the ticket type to be used in the new ticket.

'Service' or 'ServiceID' specifies the name or id of the service to be used in the new ticket.

'SLA' or 'SLAID' specifies the name or id of the SLA to be used in the new ticket.

'ResponsibleID', the ID of the agent that will be the new ticket responsible.

'PendingTime', a predefined date to set the Ticket Pending Times, when the ticket state belongs to a pending state type.

'PendingTimeDiff', a dynamically date (expressed in seconds from current date/time) to set the Ticket Pending Times, when the ticket state belongs to a pending state type.

'ArticleType' defines the type of the article to be created. Possible values: phone, fax, sms, webrequest, note-internal, note-external and note-report.

SenderType defines the sender type of the article. Possible values: agent, system, customer.

'ContentType' defines the content type of the article. Possible values: 'text/plain; charset=ISO-8859-15' or any other valid charset and mime type.

'Subject' defines the article title. Mandatory.

'Body' defines the article content. Mandatory.

'HistoryType' defines the type of the history entry. Possible values: AddNote, ArchiveFlagUpdate, Bounce, CustomerUpdate, EmailAgent, EmailCustomer, EscalationResponseTimeNotifyBefore, EscalationResponseTimeStart, EscalationResponseTimeStop, EscalationSolutionTimeNotifyBefore, EscalationSolutionTimeStart, EscalationSolutionTimeStop, EscalationUpdateTimeNotifyBefore, EscalationUpdateTimeStart, EscalationUpdateTimeStop, FollowUp, Forward, Lock, LoopProtection, Merged, Misc, Move, NewTicket, OwnerUpdate, PhoneCallAgent, PhoneCallCustomer, PriorityUpdate, Remove, ResponsibleUpdate, SendAgentNotification, SendAnswer, SendAutoFollowUp, SendAutoReject, SendAutoReply, SendCustomerNotification, ServiceUpdate, SetPendingTime, SLAUpdate, StateUpdate, Subscribe, SystemRequest, TicketDynamicFieldUpdate, TicketLinkAdd, TicketLinkDelete, TimeAccounting, TypeUpdate, Unlock, Unsubscribe, WebRequestCustomer.

'HistoryComment' defines the content of the history entry.

'From', 'To', 'Cc' and 'ReplyTo' take email addresses in the notation specified above.

'InReplyTo' and 'References' take email message IDs.

'NoAgentNotify' - if set to 1, the email notification of the Agent will not be sent.

'AutoResponseType' can take the following values: auto follow up, auto reject, auto remove, auto reply, auto reply/new ticket.

'ForceNotificationToUserID', 'ExcludeNotificationToUserID', 'ExcludeMuteNotificationToUserID' can take a list of UserIDs that are either always notified, not notified or listed as notified but not actually sent a notification email.

'TimeUnit' the time invested in the current ticket article expressed in seconds, minutes, hours, etc.

'DynamicField_NameX' where DynamicField_ is a required prefix and NameX is the name of a Dynamic Field to be set in the new ticket (on ticket level, not article levels).

'LinkAs' to define the new ticket relation with originator ticket, from the new ticket point of view, for example Normal, Parent, Child etc.

2.4.5.6.4. TicketCustomerSet

Sets the customer of a process ticket. Example:

```
$Self->{'Process::TransitionAction'} = {
    'TA1' => {
        Name   => 'Customer Set Customer to test',
        Module => 'Kernel::System::Process::TransitionAction::TicketCustomerSet',
        Config => {
            No      => 'test',
            User    => 'client-user-123',
            # or in other words
            # CustomerID     => 'client123',
            # CustomerUserID => 'client-user-123',

        },
    },
};
```

'Name' specifies the name of the configured TransitionAction.

'No' or 'CustomerID' set the Customer ID of the customer.

'User' or 'CustomerUserID' set the Username of the customer.

2.4.5.6.5. TicketLockSet

Changes the lock of a process ticket. Example:

```
$Self->{'Process::TransitionAction'} = {
    'TA1' => {
        Name   => 'Set Lock to lock',
        Module => 'Kernel::System::ProcessManagement::TransitionAction::TicketLockSet',
        Config => {
            Lock   => 'lock',
            # or
            LockID => 2,
        },
    },
};
```

'Name' specifies the name of the configured TransitionAction.

'Lock' defines the new lock of the process ticket.

'LockID' defines the internal ID of the new lock.

2.4.5.6.6. TicketOwnerSet

Changes the owner of a process ticket. Example:

```
$Self->{'Process::TransitionAction'} = {
    'TA1' => {
        Name   => 'Owner Set root@localhost',
        Module => 'Kernel::System::ProcessManagement::TransitionAction::TicketOwnerSet',
        Config => {
            Owner => 'root@localhost',
            # or
            OwnerID => 1,
        },
    },
};
```

'Name' specifies the name of the configured TransitionAction.

'Owner' specifies the login name of the new owner.

'OwnerID' specifies the internal ID of the new owner.

2.4.5.6.7. TicketQueueSet

Moves the ticket into a target queue. Example:

```
$Self->{'Process::TransitionAction'} = {
    'TA1' => {
        Name   => 'Queue Move Raw',
        Module => 'Kernel::System::ProcessManagement::TransitionAction::TicketQueueSet',
        Config => {
            Queue => 'Raw',
            # or
            # QueueID => '2',
        },
    },
};
```

'Name' specifies the name of the configured TransitionAction.

'Queue' specifies the name of the target queue.

'QueueID' specifies the internal ID of the target queue.

2.4.5.6.8. TicketResponsibleSet

Changes the responsible of a process ticket. Example:

```
$Self->{'Process::TransitionAction'} = {
    'TA1' => {
        Name   => 'Responsible Set root@localhost',
        Module =>
 'Kernel::System::ProcessManagement::TransitionAction::TicketResponsibleSet',
        Config => {
            Responsible => 'root@localhost',
            # or
            ResponsibleID => 1,
        },
    },
};
```

'Name' specifies the name of the configured TransitionAction.

'Responsible' specifies the login name of the new responsible.

'ResponsibleID' specifies the internal ID of the new responsible.

2.4.5.6.9. TicketServiceSet

Assigns a service to a process ticket. The ticket requires to have a customer and the service must be assigned to that customer. Example:

```
$Self->{'Process::TransitionAction'} = {
    'TA1' => {
        Name   => 'Set MyService service',
        Module => 'Kernel::System::ProcessManagement::TransitionAction::TicketServiceSet',
        Config => {
            Service   => 'MyService',
            # or
            ServiceID => 123,
        },
    },
};
```

'Name' specifies the name of the configured TransitionAction.

'Service' defines the new service of the process ticket. The full name is required (e.g. GramdFatherService::FatherService::SonService).

'ServiceID' defines the internal ID of the new service.

2.4.5.6.10. TicketSLASet

Assigns a service level agreement to a process ticket. The ticket requires to have a service and the SLA must be assigned to that service. Example:

```
$Self->{'Process::TransitionAction'} = {
    'TA1' => {
        Name   => 'Set MySLA SLA',
        Module => 'Kernel::System::ProcessManagement::TransitionAction::TicketSLASet',
        Config => {
            SLA   => 'MySLA',
            # or
            SLAID => 123,
        },
    },
};
```

'Name' specifies the name of the configured TransitionAction.

'SLA' defines the new service level agreement of the process ticket.

'SLAID' defines the internal ID of the new SLA.

2.4.5.6.11. TicketStateSet

Changes the state of a process ticket. Example:

```
$Self->{'Process::TransitionAction'} = {
    'TA1' => {
        Name   => 'Set State to open',
        Module => 'Kernel::System::ProcessManagement::TransitionAction::TicketStateSet',
        Config => {
            State   => 'open',
            # or
            StateID => 4,

            PendingTimeDiff => 123,
        },
    },
```

```
};
```

'Name' specifies the name of the configured TransitionAction.

'State' defines the new state of the process ticket.

'StateID' defines the internal ID of the new state.

'PendingTimeDiff' used only for pending type states, defines the time difference in seconds relative (relative to the Transition Action execution time) to set ticket pending time (e.g. 3600 means that the pending time is 1hr after the Transition Action is executed).

2.4.5.6.12. TicketTitleSet

Sets the ticket title of a process ticket. Example:

```
$Self->{'Process::TransitionAction'} = {
    'TA1' => {
        Name   => 'Set Ticket Title to Ticket-title',
        Module => 'Kernel::System::ProcessManagement::TransitionAction::TicketTitleSet',
        Config => {
            Title => 'Ticket-title',
        },
    },
};
```

'Name' specifies the name of the configured TransitionAction.

'Title' specifies the new title of the ticket.

2.4.5.6.13. TicketTypeSet

Sets the ticket type of a process ticket. Example:

```
$Self->{'Process::TransitionAction'} = {
    'TA1' => {
        Name   => 'Set Ticket Type to default',
        Module => 'Kernel::System::ProcessManagement::TransitionAction::TicketTypeSet',
        Config => {
            Type     => 'default',
            # or
            # TypeID => '1',
        },
    },
};
```

'Name' specifies the name of the configured TransitionAction.

'Type' specifies the name of the ticket type.

'TypeID' specifies the internal ID of the ticket type.

2.4.6. Access Control Lists (ACLs)

With the help of ACLs, you can limit selectable values in process tickets. Please also see the ACL reference for a description of the full ticket ACL syntax.

2.4.6.1. ACL configuration

ACLs can only be defined in `Kernel/Config.pm`. Example:

```
$Self->{TicketAcl}->{'001-ACL-ProcessProperties'} = {
    Properties => {
        Process => {
            ProcessEntityID        => ['P1'],
            ActivityEntityID       => ['A1'],
            ActivityDialogEntityID => ['AD1'],
        }
    },
    Possible => {
        ActivityDialog => ['AD1', 'AD3'],
    },
    PossibleNot => {
        ActivityDialog => ['AD3'],
    },
};
```

2.4.6.2. 001-ACL-ProcessProperties

Name of the ACL rule. For further information on ACL rules in general, please consult the ACL manual.

2.4.6.3. Process

This is the section that is used to check if an ACL must be applied. If it has the specified values, the rule is applied. The following values can be used:

2.4.6.3.1. ProcessEntityID

The ID of a process that the process. Matches if the ticket is assigned to this process.

2.4.6.3.2. ActivityEntityID

The ID of the Activity that the process ticket currently is assigned to.

2.4.6.3.3. ActivityDialogEntityID

The ID of the Activity Dialog that is currently open for a process ticket.

2.4.6.4. Possible/PossibleNot Activity Dialog

Here you can specify a list of Activity Dialog IDs. This list will limit the possible Activity Dialogs that are offered to the user in the ticket zoom mask.

'Possible' lists the Activity Dialogs that are allowed. The setting above will only allow 'AD1' and 'AD3' of the list of configured Activity Dialogs.

'PossibleNot' lists the Activity Dialogs that are not allowed. In the example above, the setting will remove 'AD3' from the list of configured Activity Dialogs.

If both 'Possible' and 'PossibleNot' are specified, the list of configured Activity Dialogs will first be filtered by 'Possible', leaving only 'AD1' and 'AD3' in our example. Then 'PossibleNot' will be applied and filter out 'AD3', so that only 'AD1' remains and is shown as a possible Activity Dialog that the user can use.

If multiple ACL rules match, the intersection of all matching rules will be calculated to determine the possible Activity Dialogs. Example:

Configured Activity Dialogs: 'AD1', 'AD2', 'AD3', 'AD4', 'AD5', 'AD6', 'AD7'.

```
$Self->{TicketAcl}->{'001-ACL-Status'} = {
```

```
    Properties => {
        Ticket => {
            Status => 'new',
        }
    },
    Possible => {
        ActivityDialog => ['AD1', 'AD2', 'AD3', 'AD6', 'AD7'],
    },
};
$Self->{TicketAcl}->{'002-ACL-Queue'} = {
    Properties => {
        Ticket => {
            Queue => ['Raw']
        }
    },
    Possible => {
        ActivityDialog => ['AD2', 'AD3', 'AD4', 'AD7'],
    },
};
$Self->{TicketAcl}->{'003-ACL-Priority'} = {
    Properties => {
        Ticket => {
            Priority => ['3 normal']
        }
    },
    PossibleNot => {
        ActivityDialog => ['AD3', 'AD4'],
    },
};
```

If a process ticket has the state 'new', is in the 'Raw' queue and has a priority '3 normal', then all three ACL rules will match.

The first rule reduces the Activity Dialogs from 'AD1', 'AD2', 'AD3', 'AD4', 'AD5', 'AD6', 'AD7' to 'AD1', 'AD2', 'AD3', 'AD6', 'AD7' and forbids 'AD4' and 'AD5'.

The second rule will now further reduce the remaining Activity Dialogs. In our example, 'AD2', 'AD3', 'AD7' will remain.

Now the third rule will further reduce the list by 'PossibleNot'. 'AD3' is removed from the list. 'AD4' is not removed, since it was not on the list in the first place. At the end, 'AD2' and 'AD7' remain as possible Activity Dialogs that the user can utilize.

It is also possible to limit the processes that can be displayed in the "New process ticket" screen. The functionality is similar to limiting the Activity Dialogs with one exception: The ACLs could only be based on Users.

See examples below:

```
$Self->{TicketAcl}->{'200-ACL-Process'} = {
    # match properties
    Properties => {
        User => {
            UserID => [2, 3],
        },
    },
    Possible => {
       Process => ['P1', 'P2', 'P3'],
    },
    PossibleNot => {
        Process => ['P4'],
    },
};
```

```
$Self->{TicketAcl}->{'201-ACL-Process'} = {
```

```
    # match properties
    Properties => {
        User => {
            Group_rw => [ 'MyGroup' ],
        },
    },
    Possible => {
       Process => ['P1', 'P2', 'P3'],
    },
    PossibleNot => {
        Process => ['P4'],
    },
};
```

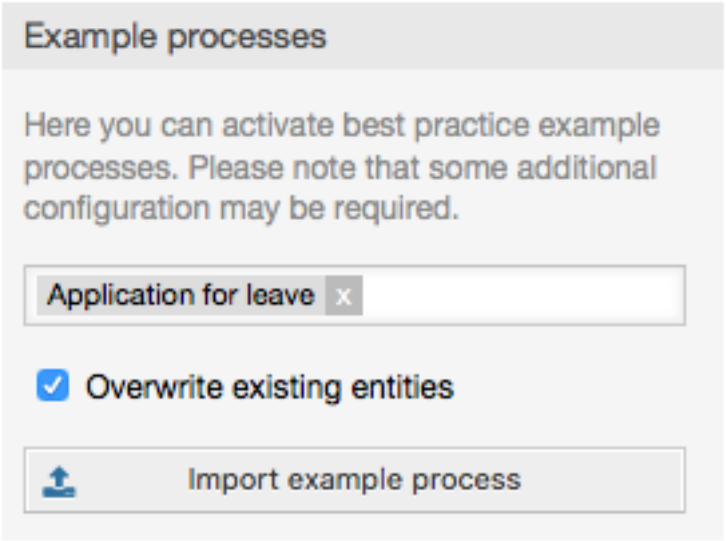
```
$Self->{TicketAcl}->{'202-ACL-Process'} = {
    # match properties
    Properties => {
        User => {
            Role => [ 'MyRole' ],
        },
    },
    Possible => {
       Process => ['P1', 'P2', 'P3'],
    },
    PossibleNot => {
        Process => ['P4'],
    },
};
```

2.5. Import example process

2.5.1. Import

On the *AdminProcessManagement* screen you can find an *Example process* widget, where you can find best practice example processes. Currently, there is only *Application for leave* process available, but you can find additional example processes in the **OTRS Business Solution™**.

Figure 5.24. Import example process widget



Select process from the drop-down menu and click on the *Import example process* button. After the process is imported, don't forget to deploy changes.

3. Creating Your Own Themes

You can create your own themes so as to use the layout you like in the OTRS web frontend. To create own themes, you should customize the output templates to your needs.

More information on the syntax and structure of output templates can be found in the Developer Manual at http://otrs.github.io/doc, especially in the chapter on *templates*.

As an example, perform the following steps to create a new theme called "Company":

1. Create a directory called `Kernel/Output/HTML/Templates/Company` and copy all files that you like to change, from `Kernel/Output/HTML/Templates/Standard` into the new folder.

   Important

   Only copy over the files you actually change. OTRS will automatically get the missing files from the Standard theme. This will make upgrading at a later stage much easier.

2. Customize the files in the directory `Kernel/Output/HTML/Templates/Company`, and change the layout to your needs.

3. To activate the new theme, add them in SysConfig under Frontend::Themes.

Now the new theme should be usable. You can select it via your personal preferences page.

Warning

Do not change the theme files shipped with OTRS, since these changes will be lost after an update. Create your own themes only by performing the steps described above.

4. Localization of the OTRS Front End

Procedures for localization for the OTRS framework, steps to be followed to create a new language translation, as well as procedures for translation customizations, can be found in the "Language Translations" chapter from the developer manual on http://otrs.github.io/doc.

Chapter 6. Performance Tuning

Presented below is a list of performance enhancing techniques for your OTRS installation, including configuration, coding, memory use, and more.

1. OTRS

There are several options for improving OTRS performance.

1.1. TicketIndexModule

There are two backend modules for the index for the ticket queue view:

- Using `Kernel::System::Ticket::IndexAccelerator::RuntimeDB` (default), generate each queue view on the fly from the ticket table. You will not have performance trouble until you have about 60,000 open tickets in your system.
- `Kernel::System::Ticket::IndexAccelerator::StaticDB`, the most powerful module, should be used when you have above 80,000 open tickets. It uses an extra ticket_index table, which works like a view. Use `bin/otrs.Console.pl Maint::Ticket::QueueIndexRebuild` for generating an initial index after switching backends.

You can change the IndexAccelerator via SysConfig.

1.2. TicketStorageModule

There are two different backend modules for the ticket/article storage:

- Configure `Kernel::System::Ticket::ArticleStorageDB` (default) to store attachments, etc., in the database. Note: Don't use it with large setups.

  Pro: If your webserver user isn't the 'otrs' user, use this module to avoid file permission problems.

  Con: It is not advisable to store attachments in your database. Take care that your database is able to store large objects. E.g. Configure MySQL with "set-variable = max_allowed_packet=8M" to store 8 MB objects (the default is 2M).

- Configure `Kernel::System::Ticket::ArticleStorageFS` to store attachments etc. on the local file system. Note: Recommended for large setups.

  Pro: It is fast!

  Con: Your web server user should be the 'otrs' user. Also, if you have multiple front-end servers, you should make sure the filesystem is shared between the servers. Place it on an NFS share or preferably a SAN or similar solution.

Note: you can switch from one back-end to the other on the fly. You can switch the backend in the SysConfig, and then run the command line utility `bin/otrs.Console.pl Admin::Article::StorageSwitch` to put the articles from the database onto the filesystem or the other way around. You can use the *--target* option to specify the target backend. Please note that the entire process can take considerable time to run, depending on the number of articles you have and the available CPU power and/or network capacity.

```
shell> bin/otrs.Console.pl Admin::Article::StorageSwitch --target ArticleStorageFS
```

Script: Switching storage back-ends from database to filesystem.

If you want to keep old attachments in the database, you can activate the SysConfig option `Ticket::StorageModule::CheckAllBackends` to make sure OTRS will still find them.

1.3. Archiving Tickets

As OTRS can be used as an audit-proof system, deleting closed tickets may not be a good idea. Therefore we implemented a feature that allows you to archive tickets.

Tickets that match certain criteria can be marked as "archived". These tickets are not accessed if you do a regular ticket search or run a Generic Agent job. The system itself does not have to deal with a huge amount of tickets any longer as only the "latest" tickets are taken into consideration when using OTRS. This can result in a huge performance gain on large systems.

To use the archive feature simply follow these steps:

1. Activate the archive system in SysConfig

   In the Admin page, go to SysConfig and select the group `Ticket`. In `Core::Ticket` you find the option `Ticket::ArchiveSystem` which is set to "no" by default. Change this setting to "yes" and save this change.

2. Define a GenericAgent job

   On the Admin page, select GenericAgent and add a new job there.

   a. Job Settings

      Provide a name for the archiving job, and select proper options to schedule this job.

   b. Ticket Filter

      The ticket filter is searches for tickets that match the selected criteria. It might be a good idea to only archive those tickets in a closed state that have been closed a few months before.

   c. Ticket Action

      In this section, set the field labeled "Archive selected tickets" to "archive tickets".

   d. Save the job

      At the end of the page you will find an option to save the job.

   e. Affected tickets

      The system will display all tickets which will be archived when executing the Generic Agent job.

3. Ticket Search

   When you search for tickets, the system default is to search tickets which are not archived. If you want to search through archived tickets also, simply add "archive search" while defining search criteria.

1.4. Cache

OTRS caches a lot of temporary data in `/opt/otrs/var/tmp`. Please make sure that this uses a high performance file system/storage. If you have enough RAM, you can also try to put this directory on a ramdisk like this:

```
shell> /opt/otrs/bin/otrs.Console.pl Maint::Session::DeleteAll
shell> /opt/otrs/bin/otrs.Console.pl Maint::Cache::Delete
shell> sudo mount -o size=16G -t tmpfs none /opt/otrs/var/tmp

# add persistent mount point in /etc/fstab
```

Note

Please note that this will be a non-permanent storage that will be lost on server reboot. All your sessions (if you store them in the filesystem) and your cache data will be lost.

There is also a centralized memcached based Cache backend available for purchase from OTRS Group.

2. Database

DB issues vary by the database being used. Study the documentation for your database or check with your database administrator.

2.1. MySQL

If you use the MySQL table type MyISAM (which is the default), and have deleted a large part of a table or if you have made many changes to a table with variable-length rows (tables that have VARCHAR, BLOB or TEXT columns), you must defragment the datafile (tables) with the "optimize" command.

You should try this if the mysqld daemon needs a lot of your CPU time. Optimize the tables - ticket, ticket_history and article (see Script below).

```
shell> mysql -u user -p database
mysql> optimize table ticket;
mysql> optimize table ticket_history;
mysql> optimize table article;
```

Script: Optimizing data base tables.

2.2. PostgreSQL

PostgreSQL is best tuned by modifying the `postgresql.conf` file in your PostgreSQL data directory. For advice on how to do this, reference the following articles:

- http://www.revsys.com/writings/postgresql-performance.html
- http://varlena.com/GeneralBits/Tidbits/perf.html
- http://varlena.com/GeneralBits/Tidbits/annotated_conf_e.html

If performance is still not satisfactory, we suggest that you join the PostgreSQL Performance mailing list (http://www.postgresql.org/community/lists/), and ask questions there. The folks on the PostgreSQL list are very friendly and can probably help.

3. Webserver

Of course you should use mod_perl 2.0 (http://perl.apache.org/). It's much faster (~ * 100) than pure cgi. But it needs more RAM.

3.1. Pre-established database connections

You can have the database connections pre-established on startup of the web server. This saves time (see README.webserver).

3.2. Preloaded modules - startup.pl

Use the startup script `scripts/apache2-perl-startup.pl` for preloaded/precompiled Perl modules on your mod_perl webserver to be faster, with a smaller memory footprint (see README.webserver).

3.3. Reload Perl modules when updated on disk

By default Apache::Reload is used in `scripts/apache2-httpd.include.conf`. Disable it and you will get 8% more speed. But remember to restart the web server if you install any modules via the OTRS Package Manager, or any values in your SysConfig or in `Kernel/Config.pm`. Important: this would also mean you can't use the OTRS Package Manager via the web interface, you need to use the command line variant - `bin/otrs.PackageManager.pl`.

3.4. Choosing the Right Strategy

If you have a larger installation, e.g. over 1,000 new tickets per day and over 40 agents, it is a good idea to read the chapters on Performance of the mod_perl User's Guide (http://perl.apache.org/docs/2.0/user/index.html).

3.5. mod_gzip/mod_deflate

If your bandwidth is small, use mod_deflate for Apache2. If you have an html page with 45k, mod_gzip/mod_deflate compresses it to about 7k. The drawback is that this increases the load on the server side.

Appendix A. Additional Resources

otrs.com

The OTRS website with source code, documentation and news is available at www.otrs.com. Here you can also find information about professional services and OTRS Administrator training seminars from OTRS Group, the creator of OTRS.

Mailing Lists

Table A.1. Mailing Lists

Name & URL	Description
announce@otrs.org	Low traffic list, in English, for announcements of new OTRS releases and security issues.
otrs@otrs.org	Medium to high traffic list, in English, where you can find all sorts of relevant questions and support for the product.
otrs-de@otrs.org	Medium to high traffic list, in German, where you can find all sorts of relevant questions and support for the product.
dev@otrs.org	Medium traffic list, in English, where the OTRS developers discuss various design and implementation issues.

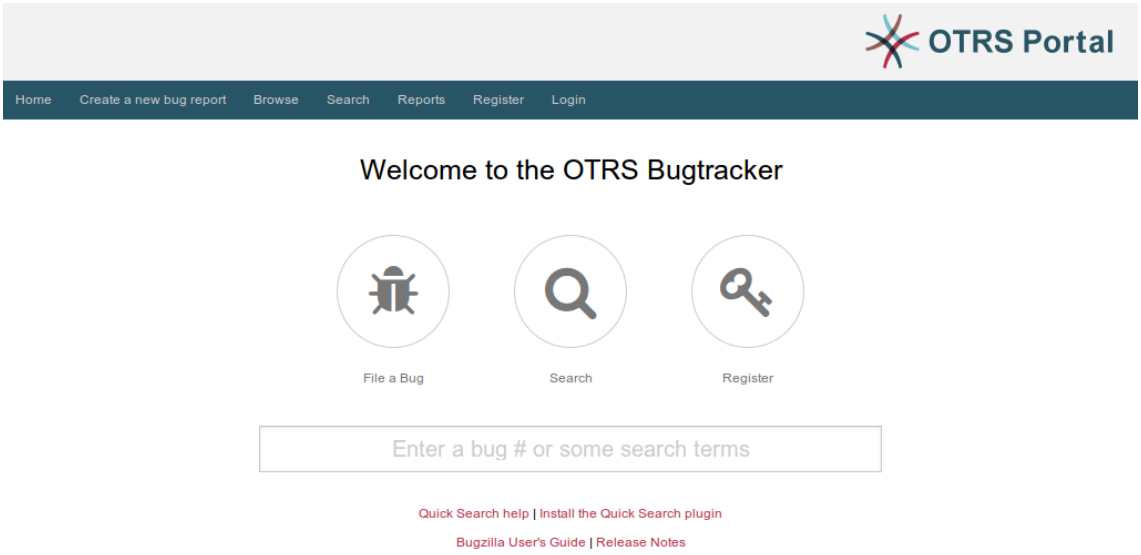
Translations

You can help translate OTRS to your language at Transifex.

Bug tracking

To report software defects, please visit http://bugs.otrs.org/ (see figure below). Please take note of the difference between a bug and a configuration issue. Configuration issues are problems that you encounter when setting a system, or general questions regarding the use of OTRS. Bug reports should only be used for issues with the source code of OTRS or other open source OTRS modules itself. For configuration issues, you should either use the commercial support, available from OTRS, or the public mailing lists.

Figure A.1. Bugtracking Tool



Appendix B. Configuration Options Reference

1. CloudService

CloudService → CloudService::Admin::ModuleRegistration

CloudService::Admin::Module###100-SupportDataCollector
Cloud service admin module registration for the transport layer.

Default value:

```
$Self->{'CloudService::Admin::Module'}->{'100-SupportDataCollector'} =  {
  'ConfigDialog' => 'AdminCloudServiceSupportDataCollector',
  'Description' => 'Configure sending of support data to OTRS Group for improved
 support.',
  'Icon' => 'fa fa-compass',
  'Name' => 'Support data collector'
};
```

CloudService::Admin::Module###200-SMS
Cloud service admin module registration for the transport layer.

Default value:

```
$Self->{'CloudService::Admin::Module'}->{'200-SMS'} =  {
  'ConfigDialog' => 'AdminCloudServiceSMS',
  'Description' => 'This will allow the system to send text messages via SMS.',
  'Icon' => 'fa fa-mobile',
  'IsOTRSBusiness' => '1',
  'Name' => 'SMS'
};
```

CloudService → Core

CloudServices::Disabled
Disables the communication between this system and OTRS Group servers that provides cloud services. If active, some functionality will be lost such as system registration, support data sending, upgrading to and use of OTRS Business Solution™, OTRS Verify™, OTRS News and product News dashboard widgets, among others.

This setting is not active by default.

Default value:

```
$Self->{'CloudServices::Disabled'} =  '0';
```

CloudService → Frontend::Agent::ModuleNotify

Frontend::NotifyModule###100-CloudServicesDisabled
Defines the module to display a notification if cloud services are disabled.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::NotifyModule'}->{'100-CloudServicesDisabled'} =  {
  'Group' => 'admin',
  'Module' => 'Kernel::Output::HTML::Notification::AgentCloudServicesDisabled'
```

```
};
```

2. Daemon

Daemon → Core::Daemon::ModuleRegistration

DaemonModules###SchedulerGenericAgentTaskManager
The daemon registration for the scheduler generic agent task manager.

This setting can not be deactivated.

Default value:

```
$Self->{'DaemonModules'}->{'SchedulerGenericAgentTaskManager'} =  {
  'Module' => 'Kernel::System::Daemon::DaemonModules::SchedulerGenericAgentTaskManager'
};
```

DaemonModules###SchedulerCronTaskManager
The daemon registration for the scheduler cron task manager.

This setting can not be deactivated.

Default value:

```
$Self->{'DaemonModules'}->{'SchedulerCronTaskManager'} =  {
  'Module' => 'Kernel::System::Daemon::DaemonModules::SchedulerCronTaskManager'
};
```

DaemonModules###SchedulerFutureTaskManager
The daemon registration for the scheduler future task manager.

This setting can not be deactivated.

Default value:

```
$Self->{'DaemonModules'}->{'SchedulerFutureTaskManager'} =  {
  'Module' => 'Kernel::System::Daemon::DaemonModules::SchedulerFutureTaskManager'
};
```

DaemonModules###SchedulerTaskWorker
The daemon registration for the scheduler task worker.

This setting can not be deactivated.

Default value:

```
$Self->{'DaemonModules'}->{'SchedulerTaskWorker'} =  {
  'Module' => 'Kernel::System::Daemon::DaemonModules::SchedulerTaskWorker'
};
```

Daemon → Core::Log

Daemon::Log::DaysToKeep
Defines the number of days to keep the daemon log files.

This setting is not active by default.

Default value:

```
$Self->{'Daemon::Log::DaysToKeep'} =  '1';
```

Daemon::Log::STDOUT
If enabled the daemon will redirect the standard output stream to a log file.

This setting can not be deactivated.

Default value:

```
$Self->{'Daemon::Log::STDOUT'} =  '0';
```

Daemon::Log::STDERR
If enabled the daemon will redirect the standard error stream to a log file.

This setting can not be deactivated.

Default value:

```
$Self->{'Daemon::Log::STDERR'} =  '1';
```

Daemon → Core::Web

Loader::Agent::CommonCSS###001-Daemon
List of CSS files to always be loaded for the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Loader::Agent::CommonCSS'}->{'001-Daemon'} =  [
  'Core.Agent.DaemonInfo.css'
];
```

Loader::Agent::CommonJS###001-Daemon
List of JS files to always be loaded for the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Loader::Agent::CommonJS'}->{'001-Daemon'} =  [
  'Core.Agent.DaemonInfo.js'
];
```

Daemon → Daemon::SchedulerCronTaskManager::Task

Daemon::SchedulerCronTaskManager::Task###CoreCacheCleanup
Delete expired cache from core modules.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'CoreCacheCleanup'} =  {
  'Function' => 'CleanUp',
  'MaximumParallelInstances' => '1',
  'Module' => 'Kernel::System::Cache',
  'Params' => [
    'Expired',
    '1'
  ],
  'Schedule' => '20 0 * * 0',
  'TaskName' => 'CoreCacheCleanup'
};
```

Daemon::SchedulerCronTaskManager::Task###LoaderCacheDelete
Delete expired loader cache weekly (Sunday mornings).

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'LoaderCacheDelete'} =  {
  'Function' => 'CacheDelete',
```

```
  'MaximumParallelInstances' => '1',
  'Module' => 'Kernel::System::Loader',
  'Params' => [],
  'Schedule' => '30 0 * * 0',
  'TaskName' => 'LoaderCacheDelete'
};
```

Daemon::SchedulerCronTaskManager::Task###FetchMail
Fetch emails via fetchmail.

This setting is not active by default.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'FetchMail'} =  {
  'Function' => 'Fetch',
  'MaximumParallelInstances' => '1',
  'Module' => 'Kernel::System::FetchMail',
  'Params' => [],
  'Schedule' => '*/5 * * * *',
  'TaskName' => 'FetchMail'
};
```

Daemon::SchedulerCronTaskManager::Task###FetchMailSSL
Fetch emails via fetchmail (using SSL).

This setting is not active by default.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'FetchMailSSL'} =  {
  'Function' => 'Fetch',
  'MaximumParallelInstances' => '1',
  'Module' => 'Kernel::System::FetchMail',
  'Params' => [
    'SSL',
    '1'
  ],
  'Schedule' => '*/5 * * * *',
  'TaskName' => 'FetchMailSSL'
};
```

Daemon::SchedulerCronTaskManager::Task###GenerateDashboardStats
Generate dashboard statistics.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'GenerateDashboardStats'} =  {
  'Function' => 'Execute',
  'MaximumParallelInstances' => '1',
  'Module' => 'Kernel::System::Console::Command::Maint::Stats::Dashboard::Generate',
  'Params' => [],
  'Schedule' => '5 * * * *',
  'TaskName' => 'GenerateDashboardStats'
};
```

Daemon::SchedulerCronTaskManager::Task###EscalationCheck
Triggers ticket escalation events and notification events for escalation.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'EscalationCheck'} =  {
  'Function' => 'Execute',
  'MaximumParallelInstances' => '1',
  'Module' => 'Kernel::System::Console::Command::Maint::Ticket::EscalationCheck',
  'Params' => [],
  'Schedule' => '*/5 * * * *',
  'TaskName' => 'EscalationCheck'
};
```

Daemon::SchedulerCronTaskManager::Task###TicketPendingCheck
Process pending tickets.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'TicketPendingCheck'} =  {
  'Function' => 'Execute',
  'MaximumParallelInstances' => '1',
  'Module' => 'Kernel::System::Console::Command::Maint::Ticket::PendingCheck',
  'Params' => [],
  'Schedule' => '45 */2 * * *',
  'TaskName' => 'TicketPendingCheck'
};
```

Daemon::SchedulerCronTaskManager::Task###SpoolMailsReprocess
Reprocess mails from spool directory that could not be imported in the first place.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'SpoolMailsReprocess'} =  {
  'Function' => 'Execute',
  'MaximumParallelInstances' => '1',
  'Module' =>
 'Kernel::System::Console::Command::Maint::PostMaster::SpoolMailsReprocess',
  'Params' => [],
  'Schedule' => '10 0 * * *',
  'TaskName' => 'SpoolMailsReprocess'
};
```

Daemon::SchedulerCronTaskManager::Task###MailAccountFetch
Fetch incoming emails from configured mail accounts.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'MailAccountFetch'} =  {
  'Function' => 'Execute',
  'MaximumParallelInstances' => '1',
  'Module' => 'Kernel::System::Console::Command::Maint::PostMaster::MailAccountFetch',
  'Params' => [],
  'Schedule' => '*/10 * * * *',
  'TaskName' => 'MailAccountFetch'
};
```

Daemon::SchedulerCronTaskManager::Task###TicketAcceleratorRebuild
Rebuild the ticket index for AgentTicketQueue.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'TicketAcceleratorRebuild'} =  {
  'Function' => 'TicketAcceleratorRebuild',
  'MaximumParallelInstances' => '1',
  'Module' => 'Kernel::System::Ticket',
  'Params' => [],
  'Schedule' => '01 01 * * *',
  'TaskName' => 'TicketAcceleratorRebuild'
};
```

Daemon::SchedulerCronTaskManager::Task###SessionDeleteExpired
Delete expired sessions.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'SessionDeleteExpired'} =  {
  'Function' => 'Execute',
  'MaximumParallelInstances' => '1',
  'Module' => 'Kernel::System::Console::Command::Maint::Session::DeleteExpired',
  'Params' => [],
  'Schedule' => '55 */2 * * *',
  'TaskName' => 'SessionDeleteExpired'
```

```
};
```

Daemon::SchedulerCronTaskManager::Task###TicketUnlockTimeout
Unlock tickets that are past their unlock timeout.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'TicketUnlockTimeout'} =  {
  'Function' => 'Execute',
  'MaximumParallelInstances' => '1',
  'Module' => 'Kernel::System::Console::Command::Maint::Ticket::UnlockTimeout',
  'Params' => [],
  'Schedule' => '35 * * * *',
  'TaskName' => 'TicketUnlockTimeout'
};
```

Daemon::SchedulerCronTaskManager::Task###RenewCustomerSMIMECertificates
Renew existing SMIME certificates from customer backend. Note: SMIME and SMIME::FetchFromCustomer needs to be active in SysConfig and customer backend needs to be configured to fetch UserSMIMECertificate attribute.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'RenewCustomerSMIMECertificates'} =
  {
  'Function' => 'Execute',
  'MaximumParallelInstances' => '1',
  'Module' =>
 'Kernel::System::Console::Command::Maint::SMIME::CustomerCertificate::Renew',
  'Params' => [],
  'Schedule' => '02 02 * * *',
  'TaskName' => 'RenewCustomerSMIMECertificates'
};
```

Daemon::SchedulerCronTaskManager::Task###Custom1
Executes a custom command or module. Note: if module is used, function is required.

This setting is not active by default.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'Custom1'} =  {
  'Function' => '',
  'MaximumParallelInstances' => '1',
  'Module' => '',
  'Params' => [],
  'Schedule' => '* * * * *',
  'TaskName' => 'Custom1'
};
```

Daemon::SchedulerCronTaskManager::Task###Custom2
Executes a custom command or module. Note: if module is used, function is required.

This setting is not active by default.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'Custom2'} =  {
  'Function' => '',
  'MaximumParallelInstances' => '1',
  'Module' => '',
  'Params' => [],
  'Schedule' => '* * * * *',
  'TaskName' => 'Custom2'
};
```

Daemon::SchedulerCronTaskManager::Task###Custom3
Executes a custom command or module. Note: if module is used, function is required.

This setting is not active by default.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'Custom3'} =  {
  'Function' => '',
  'MaximumParallelInstances' => '1',
  'Module' => '',
  'Params' => [],
  'Schedule' => '* * * * *',
  'TaskName' => 'Custom3'
};
```

Daemon::SchedulerCronTaskManager::Task###Custom4

Executes a custom command or module. Note: if module is used, function is required.

This setting is not active by default.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'Custom4'} =  {
  'Function' => '',
  'MaximumParallelInstances' => '1',
  'Module' => '',
  'Params' => [],
  'Schedule' => '* * * * *',
  'TaskName' => 'Custom4'
};
```

Daemon::SchedulerCronTaskManager::Task###Custom5

Executes a custom command or module. Note: if module is used, function is required.

This setting is not active by default.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'Custom5'} =  {
  'Function' => '',
  'MaximumParallelInstances' => '1',
  'Module' => '',
  'Params' => [],
  'Schedule' => '* * * * *',
  'TaskName' => 'Custom5'
};
```

Daemon::SchedulerCronTaskManager::Task###Custom6

Executes a custom command or module. Note: if module is used, function is required.

This setting is not active by default.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'Custom6'} =  {
  'Function' => '',
  'MaximumParallelInstances' => '1',
  'Module' => '',
  'Params' => [],
  'Schedule' => '* * * * *',
  'TaskName' => 'Custom6'
};
```

Daemon::SchedulerCronTaskManager::Task###Custom7

Executes a custom command or module. Note: if module is used, function is required.

This setting is not active by default.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'Custom7'} =  {
  'Function' => '',
  'MaximumParallelInstances' => '1',
  'Module' => '',
  'Params' => [],
  'Schedule' => '* * * * *',
  'TaskName' => 'Custom7'
};
```

Daemon::SchedulerCronTaskManager::Task###Custom8

Executes a custom command or module. Note: if module is used, function is required.

This setting is not active by default.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'Custom8'} =  {
  'Function' => '',
  'MaximumParallelInstances' => '1',
  'Module' => '',
  'Params' => [],
  'Schedule' => '* * * * *',
  'TaskName' => 'Custom8'
};
```

Daemon::SchedulerCronTaskManager::Task###Custom9

Executes a custom command or module. Note: if module is used, function is required.

This setting is not active by default.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'Custom9'} =  {
  'Function' => '',
  'MaximumParallelInstances' => '1',
  'Module' => '',
  'Params' => [],
  'Schedule' => '* * * * *',
  'TaskName' => 'Custom9'
};
```

Daemon::SchedulerCronTaskManager::Task###GenericAgentFile1

Run file based generic agent jobs (Note: module name need needs to be specified in -configuration-module param e.g. "Kernel::System::GenericAgent").

This setting is not active by default.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'GenericAgentFile1'} =  {
  'Function' => 'Execute',
  'MaximumParallelInstances' => '1',
  'Module' => 'Kernel::System::Console::Command::Maint::GenericAgent::Run',
  'Params' => [
    '--configuration-module',
    '<ModuleName>'
  ],
  'Schedule' => '*/20 * * * *',
  'TaskName' => 'GenericAgentFile1'
};
```

Daemon::SchedulerCronTaskManager::Task###GenericAgentFile2

Run file based generic agent jobs (Note: module name need needs to be specified in -configuration-module param e.g. "Kernel::System::GenericAgent").

This setting is not active by default.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'GenericAgentFile2'} =  {
  'Function' => 'Execute',
  'MaximumParallelInstances' => '1',
  'Module' => 'Kernel::System::Console::Command::Maint::GenericAgent::Run',
  'Params' => [
    '--configuration-module',
    '<ModuleName>'
  ],
  'Schedule' => '*/20 * * * *',
  'TaskName' => 'GenericAgentFile2'
};
```

Daemon::SchedulerCronTaskManager::Task###GenericAgentFile3

Run file based generic agent jobs (Note: module name need needs to be specified in -configuration-module param e.g. "Kernel::System::GenericAgent").

This setting is not active by default.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'GenericAgentFile3'} =  {
  'Function' => 'Execute',
  'MaximumParallelInstances' => '1',
  'Module' => 'Kernel::System::Console::Command::Maint::GenericAgent::Run',
  'Params' => [
    '--configuration-module',
    '<ModuleName>'
  ],
  'Schedule' => '*/20 * * * *',
  'TaskName' => 'GenericAgentFile3'
};
```

Daemon::SchedulerCronTaskManager::Task###GenericAgentFile4

Run file based generic agent jobs (Note: module name need needs to be specified in -configuration-module param e.g. "Kernel::System::GenericAgent").

This setting is not active by default.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'GenericAgentFile4'} =  {
  'Function' => 'Execute',
  'MaximumParallelInstances' => '1',
  'Module' => 'Kernel::System::Console::Command::Maint::GenericAgent::Run',
  'Params' => [
    '--configuration-module',
    '<ModuleName>'
  ],
  'Schedule' => '*/20 * * * *',
  'TaskName' => 'GenericAgentFile4'
};
```

Daemon::SchedulerCronTaskManager::Task###GenericAgentFile5

Run file based generic agent jobs (Note: module name need needs to be specified in -configuration-module param e.g. "Kernel::System::GenericAgent").

This setting is not active by default.

Default value:

```
$Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'GenericAgentFile5'} =  {
  'Function' => 'Execute',
  'MaximumParallelInstances' => '1',
  'Module' => 'Kernel::System::Console::Command::Maint::GenericAgent::Run',
  'Params' => [
    '--configuration-module',
    '<ModuleName>'
  ],
  'Schedule' => '*/20 * * * *',
```

```
  'TaskName' => 'GenericAgentFile5'
};
```

Daemon::SchedulerCronTaskManager::Task###RegistrationUpdateSend
Sends registration information to OTRS group.

This setting can not be changed.

Default value:

```
 $Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'RegistrationUpdateSend'} =  {
  'Function' => 'Execute',
  'MaximumParallelInstances' => '1',
  'Module' => 'Kernel::System::Console::Command::Maint::Registration::UpdateSend',
  'Params' => [],
  'Schedule' => '30 * * * *',
  'TaskName' => 'RegistrationUpdateSend'
};
```

Daemon::SchedulerCronTaskManager::Task###SupportDataCollectAsynchronous
Collect support data for asynchronous plug-in modules.

This setting can not be changed.

Default value:

```
 $Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'SupportDataCollectAsynchronous'} =
  {
  'Function' => 'Execute',
  'MaximumParallelInstances' => '1',
  'Module' =>
 'Kernel::System::Console::Command::Maint::SupportData::CollectAsynchronous',
  'Params' => [],
  'Schedule' => '1 * * * *',
  'TaskName' => 'SupportDataCollectAsynchronous'
};
```

Daemon::SchedulerCronTaskManager::Task###OTRSBusinessEntitlementCheck
Checks the entitlement status of OTRS Business Solution™.

This setting can not be changed.

Default value:

```
 $Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'OTRSBusinessEntitlementCheck'} =
 {
  'Function' => 'Execute',
  'MaximumParallelInstances' => '1',
  'Module' => 'Kernel::System::Console::Command::Maint::OTRSBusiness::EntitlementCheck',
  'Params' => [],
  'Schedule' => '25,45 */1 * * *',
  'TaskName' => 'OTRSBusinessEntitlementCheck'
};
```

Daemon::SchedulerCronTaskManager::Task###OTRSBusinessAvailabilityCheck
Checks the availability of OTRS Business Solution™ for this system.

This setting can not be changed.

Default value:

```
 $Self->{'Daemon::SchedulerCronTaskManager::Task'}->{'OTRSBusinessAvailabilityCheck'} =
 {
  'Function' => 'Execute',
  'MaximumParallelInstances' => '1',
  'Module' =>
 'Kernel::System::Console::Command::Maint::OTRSBusiness::AvailabilityCheck',
  'Params' => [],
```

```
  'Schedule' => '15,35,55 */1 * * *',
  'TaskName' => 'OTRSBusinessAvailabilityCheck'
};
```

Daemon → Daemon::SchedulerGenericAgentTaskManager

Daemon::SchedulerGenericAgentTaskManager::TicketLimit
Defines the maximum number of affected tickets per job.

This setting can not be deactivated.

Default value:

```
$Self->{'Daemon::SchedulerGenericAgentTaskManager::TicketLimit'} =  '4000';
```

Daemon::SchedulerGenericAgentTaskManager::SleepTime
Defines a sleep time in microseconds between tickets while they are been processed by a job.

This setting can not be deactivated.

Default value:

```
$Self->{'Daemon::SchedulerGenericAgentTaskManager::SleepTime'} =  '0';
```

Daemon → Daemon::SchedulerGenericInterfaceTaskManager

Daemon::SchedulerGenericInterfaceTaskManager::FutureTaskTimeDiff
Defines the default the number of seconds (from current time) to re-schedule a generic interface failed task.

This setting can not be deactivated.

Default value:

```
$Self->{'Daemon::SchedulerGenericInterfaceTaskManager::FutureTaskTimeDiff'} =  '300';
```

Daemon → Daemon::SchedulerTaskWorker

Daemon::SchedulerTaskWorker::MaximumWorkers
Defines the maximum number of tasks to be executed as the same time.

This setting can not be deactivated.

Default value:

```
$Self->{'Daemon::SchedulerTaskWorker::MaximumWorkers'} =  '5';
```

Daemon::SchedulerTaskWorker::NotificationRecipientEmail
Specifies the email addresses to get notification messages from scheduler tasks.

This setting can not be deactivated.

Default value:

```
$Self->{'Daemon::SchedulerTaskWorker::NotificationRecipientEmail'} =  'root@localhost';
```

Daemon → Frontend::Admin::ModuleRegistration

Frontend::Module###AgentDaemonInfo
Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentDaemonInfo'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'Title' => 'Shows information on how to start OTRS Daemon'
};
```

Daemon → Frontend::Agent::ModuleNotify

Frontend::NotifyModule###800-Daemon-Check

Defines the module to display a notification in the agent interface if the OTRS Daemon is not running.

Default value:

```
$Self->{'Frontend::NotifyModule'}->{'800-Daemon-Check'} =  {
  'Module' => 'Kernel::Output::HTML::Notification::DaemonCheck'
};
```

3. DynamicFields

DynamicFields → DynamicFields::Driver::Registration

DynamicFields::Driver###Text

DynamicField backend registration.

Default value:

```
$Self->{'DynamicFields::Driver'}->{'Text'} =  {
  'ConfigDialog' => 'AdminDynamicFieldText',
  'DisplayName' => 'Text',
  'Module' => 'Kernel::System::DynamicField::Driver::Text'
};
```

DynamicFields::Driver###TextArea

DynamicField backend registration.

Default value:

```
$Self->{'DynamicFields::Driver'}->{'TextArea'} =  {
  'ConfigDialog' => 'AdminDynamicFieldText',
  'DisplayName' => 'Textarea',
  'Module' => 'Kernel::System::DynamicField::Driver::TextArea'
};
```

DynamicFields::Driver###Checkbox

DynamicField backend registration.

Default value:

```
$Self->{'DynamicFields::Driver'}->{'Checkbox'} =  {
  'ConfigDialog' => 'AdminDynamicFieldCheckbox',
  'DisplayName' => 'Checkbox',
  'Module' => 'Kernel::System::DynamicField::Driver::Checkbox'
};
```

DynamicFields::Driver###Dropdown

DynamicField backend registration.

Default value:

```
$Self->{'DynamicFields::Driver'}->{'Dropdown'} =  {
  'ConfigDialog' => 'AdminDynamicFieldDropdown',
  'DisplayName' => 'Dropdown',
```

```
  'Module' => 'Kernel::System::DynamicField::Driver::Dropdown'
};
```

DynamicFields::Driver###DateTime
DynamicField backend registration.

Default value:

```
$Self->{'DynamicFields::Driver'}->{'DateTime'} =  {
  'ConfigDialog' => 'AdminDynamicFieldDateTime',
  'DisplayName' => 'Date / Time',
  'Module' => 'Kernel::System::DynamicField::Driver::DateTime'
};
```

DynamicFields::Driver###Date
DynamicField backend registration.

Default value:

```
$Self->{'DynamicFields::Driver'}->{'Date'} =  {
  'ConfigDialog' => 'AdminDynamicFieldDateTime',
  'DisplayName' => 'Date',
  'Module' => 'Kernel::System::DynamicField::Driver::Date'
};
```

DynamicFields::Driver###Multiselect
DynamicField backend registration.

Default value:

```
$Self->{'DynamicFields::Driver'}->{'Multiselect'} =  {
  'ConfigDialog' => 'AdminDynamicFieldMultiselect',
  'DisplayName' => 'Multiselect',
  'ItemSeparator' => ', ',
  'Module' => 'Kernel::System::DynamicField::Driver::Multiselect'
};
```

DynamicFields → DynamicFields::ObjectType::Registration

DynamicFields::ObjectType###Article
DynamicField object registration.

Default value:

```
$Self->{'DynamicFields::ObjectType'}->{'Article'} =  {
  'DisplayName' => 'Article',
  'Module' => 'Kernel::System::DynamicField::ObjectType::Article',
  'Prio' => '110'
};
```

DynamicFields::ObjectType###Ticket
DynamicField object registration.

Default value:

```
$Self->{'DynamicFields::ObjectType'}->{'Ticket'} =  {
  'DisplayName' => 'Ticket',
  'Module' => 'Kernel::System::DynamicField::ObjectType::Ticket',
  'Prio' => '100'
};
```

DynamicFields → Frontend::Admin::ModuleRegistration

Frontend::Module###AdminDynamicField
Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminDynamicField'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.DynamicField.css'
    ],
    'JavaScript' => [
      'Core.Agent.Admin.DynamicField.js'
    ]
  },
  'NavBarModule' => {
    'Block' => 'Ticket',
    'Description' => 'Create and manage dynamic fields.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Dynamic Fields',
    'Prio' => '1000'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Dynamic Fields GUI'
};
```

Frontend::Module###AdminDynamicFieldText

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminDynamicFieldText'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.Admin.DynamicField.js',
      'Core.Agent.Admin.DynamicFieldText.js'
    ]
  },
  'Title' => 'Dynamic Fields Text Backend GUI'
};
```

Frontend::Module###AdminDynamicFieldCheckbox

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminDynamicFieldCheckbox'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.Admin.DynamicField.js'
    ]
  },
  'Title' => 'Dynamic Fields Checkbox Backend GUI'
};
```

Frontend::Module###AdminDynamicFieldDropdown

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminDynamicFieldDropdown'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
```

```
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.DynamicField.css'
    ],
    'JavaScript' => [
      'Core.Agent.Admin.DynamicField.js',
      'Core.Agent.Admin.DynamicFieldDropdown.js'
    ]
  },
  'Title' => 'Dynamic Fields Drop-down Backend GUI'
};
```

Frontend::Module###AdminDynamicFieldDateTime

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminDynamicFieldDateTime'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.DynamicField.css'
    ],
    'JavaScript' => [
      'Core.Agent.Admin.DynamicField.js',
      'Core.Agent.Admin.DynamicFieldDateTime.js'
    ]
  },
  'Title' => 'Dynamic Fields Date Time Backend GUI'
};
```

Frontend::Module###AdminDynamicFieldMultiselect

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminDynamicFieldMultiselect'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.DynamicField.css'
    ],
    'JavaScript' => [
      'Core.Agent.Admin.DynamicField.js',
      'Core.Agent.Admin.DynamicFieldMultiselect.js'
    ]
  },
  'Title' => 'Dynamic Fields Multiselect Backend GUI'
};
```

DynamicFields → Frontend::Agent::Preferences

PreferencesGroups###DynamicField

Defines the config parameters of this item, to be shown in the preferences view.

This setting is not active by default.

Default value:

```
$Self->{'PreferencesGroups'}->{'DynamicField'} =  {
  'Active' => '1',
  'Block' => 'Input',
```

```
  'Column' => 'Other Settings',
  'Data' => '[% Env("UserDynamicField_NameX") %]',
  'Key' => 'Default value for NameX',
  'Label' => 'NameX',
  'Module' => 'Kernel::Output::HTML::Preferences::Generic',
  'PrefKey' => 'UserDynamicField_NameX',
  'Prio' => '7000'
};
```

PreferencesGroups###DynamicFieldsOverviewPageShown
Parameters for the pages (in which the dynamic fields are shown) of the dynamic fields overview.

Default value:

```
$Self->{'PreferencesGroups'}->{'DynamicFieldsOverviewPageShown'} =  {
  'Active' => '0',
  'Column' => 'Other Settings',
  'Data' => {
    '10' => '10',
    '15' => '15',
    '20' => '20',
    '25' => '25',
    '30' => '30',
    '35' => '35'
  },
  'DataSelected' => '25',
  'Key' => 'Dynamic fields limit per page for Dynamic Fields Overview',
  'Label' => 'Dynamic Fields Overview Limit',
  'Module' => 'Kernel::Output::HTML::Preferences::Generic',
  'PrefKey' => 'AdminDynamicFieldsOverviewPageShown',
  'Prio' => '8000'
};
```

4. Framework

Framework → Core

SecureMode
Disables the web installer (http://yourhost.example.com/otrs/installer.pl), to prevent the system from being hijacked. If set to "No", the system can be reinstalled and the current basic configuration will be used to pre-populate the questions within the installer script. If not active, it also disables the GenericAgent, PackageManager and SQL Box.

This setting can not be deactivated.

Default value:

```
$Self->{'SecureMode'} =  '0';
```

Frontend::DebugMode
Enables or disables the debug mode over frontend interface.

Default value:

```
$Self->{'Frontend::DebugMode'} =  '0';
```

Frontend::AjaxDebug
Delivers extended debugging information in the frontend in case any AJAX errors occur, if enabled.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::AjaxDebug'} =  '0';
```

Frontend::TemplateCache

Enables or disables the caching for templates. WARNING: Do NOT disable template caching for production environments for it will cause a massive performance drop! This setting should only be disabled for debugging reasons!

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::TemplateCache'} =  '1';
```

ConfigLevel

Sets the configuration level of the administrator. Depending on the config level, some sysconfig options will be not shown. The config levels are in in ascending order: Expert, Advanced, Beginner. The higher the config level is (e.g. Beginner is the highest), the less likely is it that the user can accidentally configure the system in a way that it is not usable any more.

This setting can not be deactivated.

Default value:

```
$Self->{'ConfigLevel'} =  '100';
```

ConfigImportAllowed

Controls if the admin is allowed to import a saved system configuration in SysConfig.

This setting can not be deactivated.

Default value:

```
$Self->{'ConfigImportAllowed'} =  '1';
```

ProductName

Defines the name of the application, shown in the web interface, tabs and title bar of the web browser.

This setting can not be deactivated.

Default value:

```
$Self->{'ProductName'} =  'OTRS 5s';
```

SystemID

Defines the system identifier. Every ticket number and http session string contains this ID. This ensures that only tickets which belong to your system will be processed as follow-ups (useful when communicating between two instances of OTRS).

This setting can not be deactivated.

Default value:

```
$Self->{'SystemID'} =  '10';
```

FQDN

Defines the fully qualified domain name of the system. This setting is used as a variable, OTRS_CONFIG_FQDN which is found in all forms of messaging used by the application, to build links to the tickets within your system.

This setting can not be deactivated.

Default value:

```
$Self->{'FQDN'} =  'yourhost.example.com';
```

SupportDataCollector::HTTPHostname

Defines the HTTP hostname for the support data collection with the public module 'PublicSupportDataCollector' (e.g. used from the OTRS Daemon).

This setting is not active by default.

Default value:

```
$Self->{'SupportDataCollector::HTTPHostname'} =  '';
```

NodeID

Defines the cluster node identifier. This is only used in cluster configurations where there is more than one OTRS frontend system. Note: only values from 1 to 99 are allowed.

This setting is not active by default.

Default value:

```
$Self->{'NodeID'} =  '1';
```

HttpType

Defines the type of protocol, used by the web server, to serve the application. If https protocol will be used instead of plain http, it must be specified here. Since this has no affect on the web server's settings or behavior, it will not change the method of access to the application and, if it is wrong, it will not prevent you from logging into the application. This setting is only used as a variable, OTRS_CONFIG_HttpType which is found in all forms of messaging used by the application, to build links to the tickets within your system.

This setting can not be deactivated.

Default value:

```
$Self->{'HttpType'} =  'http';
```

ScriptAlias

Sets the prefix to the scripts folder on the server, as configured on the web server. This setting is used as a variable, OTRS_CONFIG_ScriptAlias which is found in all forms of messaging used by the application, to build links to the tickets within the system.

This setting can not be deactivated.

Default value:

```
$Self->{'ScriptAlias'} =  'otrs/';
```

AdminEmail

Defines the system administrator's email address. It will be displayed in the error screens of the application.

This setting can not be deactivated.

Default value:

```
$Self->{'AdminEmail'} =  'admin@example.com';
```

Organization

Company name which will be included in outgoing emails as an X-Header.

This setting can not be deactivated.

Default value:

```
$Self->{'Organization'} =  'Example Company';
```

DefaultLanguage

Defines the default front-end language. All the possible values are determined by the available language files on the system (see the next setting).

This setting can not be deactivated.

Default value:

```
$Self->{'DefaultLanguage'} =  'en';
```

DefaultUsedLanguages

Defines all the languages that are available to the application. Specify only English names of languages here.

This setting can not be deactivated.

Default value:

```
$Self->{'DefaultUsedLanguages'} =  {
 'ar_SA' => 'Arabic (Saudi Arabia)',
 'bg' => 'Bulgarian',
 'ca' => 'Catalan',
 'cs' => 'Czech',
 'da' => 'Danish',
 'de' => 'German',
 'el' => 'Greek',
 'en' => 'English (United States)',
 'en_CA' => 'English (Canada)',
 'en_GB' => 'English (United Kingdom)',
 'es' => 'Spanish',
 'es_CO' => 'Spanish (Colombia)',
 'es_MX' => 'Spanish (Mexico)',
 'et' => 'Estonian',
 'fa' => 'Persian',
 'fi' => 'Finnish',
 'fr' => 'French',
 'fr_CA' => 'French (Canada)',
 'gl' => 'Galician',
 'he' => 'Hebrew',
 'hi' => 'Hindi',
 'hr' => 'Croatian',
 'hu' => 'Hungarian',
 'id' => 'Indonesian',
 'it' => 'Italian',
 'ja' => 'Japanese',
 'lt' => 'Lithuanian',
 'lv' => 'Latvian',
 'ms' => 'Malay',
 'nb_NO' => 'Norwegian',
 'nl' => 'Nederlands',
 'pl' => 'Polish',
 'pt' => 'Portuguese',
 'pt_BR' => 'Portuguese (Brasil)',
 'ru' => 'Russian',
 'sk_SK' => 'Slovak',
 'sl' => 'Slovenian',
 'sr_Cyrl' => 'Serbian Cyrillic',
 'sr_Latn' => 'Serbian Latin',
 'sv' => 'Swedish',
 'sw' => 'Swahili',
 'th_TH' => 'Thai',
 'tr' => 'Turkish',
 'uk' => 'Ukrainian',
 'vi_VN' => 'Vietnam',
 'zh_CN' => 'Chinese (Simplified)',
 'zh_TW' => 'Chinese (Traditional)'
```

```
};
```

DefaultUsedLanguagesNative

Defines all the languages that are available to the application. Specify only native names of languages here.

This setting can not be deactivated.

Default value:

```
$Self->{'DefaultUsedLanguagesNative'} =  {
 'ar_SA' => 'العربية',
 'bg' => 'Български',
 'ca' => 'Català',
 'cs' => 'Česky',
 'da' => 'Dansk',
 'de' => 'Deutsch',
 'el' => 'Ελληνικά',
 'en' => 'English (United States)',
 'en_CA' => 'English (Canada)',
 'en_GB' => 'English (United Kingdom)',
 'es' => 'Español',
 'es_CO' => 'Español (Colombia)',
 'es_MX' => 'Español (México)',
 'et' => 'Eesti',
 'fa' => 'فارسی',
 'fi' => 'Suomi',
 'fr' => 'Français',
 'fr_CA' => 'Français (Canada)',
 'gl' => 'Galego',
 'he' => '#####',
 'hi' => '######',
 'hr' => 'Hrvatski',
 'hu' => 'Magyar',
 'id' => 'Bahasa Indonesia',
 'it' => 'Italiano',
 'ja' => '日本語',
 'lt' => 'Lietuvių kalba',
 'lv' => 'Latvijas',
 'ms' => 'Melayu',
 'nb_NO' => 'Norsk bokmål',
 'nl' => 'Nederlandse',
 'pl' => 'Polski',
 'pt' => 'Português',
 'pt_BR' => 'Português Brasileiro',
 'ru' => 'Русский',
 'sk_SK' => 'Slovenčina',
 'sl' => 'Slovenščina',
 'sr_Cyrl' => 'Српски',
 'sr_Latn' => 'Srpski',
 'sv' => 'Svenska',
 'sw' => 'Kiswahili',
 'th_TH' => '#######',
 'tr' => 'Türkçe',
 'uk' => 'Українська',
 'vi_VN' => 'Vi#t Nam',
 'zh_CN' => '简体中文',
 'zh_TW' => '正體中文'
};
```

DefaultTheme

Defines the default front-end (HTML) theme to be used by the agents and customers. If you like, you can add your own theme. Please refer the administrator manual located at http://otrs.github.io/doc/.

This setting can not be deactivated.

Default value:

```
$Self->{'DefaultTheme'} =  'Standard';
```

DefaultTheme::HostBased

It is possible to configure different themes, for example to distinguish between agents and customers, to be used on a per-domain basis within the application. Using a regular expression (regex), you can configure a Key/Content pair to match a domain. The value in "Key" should match the domain, and the value in "Content" should be a valid theme on your system. Please see the example entries for the proper form of the regex.

This setting is not active by default.

Default value:

```
$Self->{'DefaultTheme::HostBased'} =  {
  'host1\\.example\\.com' => 'SomeTheme1',
  'host2\\.example\\.com' => 'SomeTheme2'
};
```

CheckMXRecord

Makes the application check the MX record of email addresses before sending an email or submitting a telephone or email ticket.

This setting can not be deactivated.

Default value:

```
$Self->{'CheckMXRecord'} =  '1';
```

CheckMXRecord::Nameserver

Defines the address of a dedicated DNS server, if necessary, for the "CheckMXRecord" look-ups.

This setting is not active by default.

Default value:

```
$Self->{'CheckMXRecord::Nameserver'} =  'ns.example.com';
```

CheckEmailAddresses

Makes the application check the syntax of email addresses.

This setting can not be deactivated.

Default value:

```
$Self->{'CheckEmailAddresses'} =  '1';
```

CheckEmailValidAddress

Defines a regular expression that excludes some addresses from the syntax check (if "CheckEmailAddresses" is set to "Yes"). Please enter a regex in this field for email addresses, that aren't syntactically valid, but are necessary for the system (i.e. "root@localhost").

This setting can not be deactivated.

Default value:

```
$Self->{'CheckEmailValidAddress'} =  '^(root@localhost|admin@localhost)$';
```

CheckEmailInvalidAddress

Defines a regular expression that filters all email addresses that should not be used in the application.

This setting can not be deactivated.

Default value:

```
$Self->{'CheckEmailInvalidAddress'} =  '@(example)\\.(..|...)$';
```

CGILogPrefix
Specifies the text that should appear in the log file to denote a CGI script entry.

This setting can not be deactivated.

Default value:

```
$Self->{'CGILogPrefix'} =  'OTRS-CGI';
```

DemoSystem
Runs the system in "Demo" mode. If set to "Yes", agents can change preferences, such as selection of language and theme via the agent web interface. These changes are only valid for the current session. It will not be possible for agents to change their passwords.

This setting can not be deactivated.

Default value:

```
$Self->{'DemoSystem'} =  '0';
```

OutOfOfficeMessageTemplate
Defines out of office message template. Two string parameters (%s) available: end date and number of days left.

Default value:

```
$Self->{'OutOfOfficeMessageTemplate'} =  '*** out of office until %s (%s d left) ***';
```

SwitchToUser
Allows the administrators to login as other users, via the users administration panel.

This setting can not be deactivated.

Default value:

```
$Self->{'SwitchToUser'} =  '0';
```

SwitchToCustomer
Allows the administrators to login as other customers, via the customer user administration panel.

This setting can not be deactivated.

Default value:

```
$Self->{'SwitchToCustomer'} =  '0';
```

SwitchToCustomer::PermissionGroup
Specifies the group where the user needs rw permissions so that he can access the "SwitchToCustomer" feature.

This setting can not be deactivated.

Default value:

```
$Self->{'SwitchToCustomer::PermissionGroup'} =  'admin';
```

NotificationSenderName
Specifies the name that should be used by the application when sending notifications. The sender name is used to build the complete display name for the notification master (i.e. "OTRS Notifications" otrs@your.example.com).

This setting can not be deactivated.

Default value:

```
$Self->{'NotificationSenderName'} =  'OTRS Notifications';
```

NotificationSenderEmail

Specifies the email address that should be used by the application when sending notifications. The email address is used to build the complete display name for the notification master (i.e. "OTRS Notifications" otrs@your.example.com). You can use the OTRS_CONFIG_FQDN variable as set in your configuation, or choose another email address.

This setting can not be deactivated.

Default value:

```
$Self->{'NotificationSenderEmail'} =  'otrs@<OTRS_CONFIG_FQDN>';
```

System::Customer::Permission

Defines the standard permissions available for customers within the application. If more permissions are needed, you can enter them here. Permissions must be hard coded to be effective. Please ensure, when adding any of the afore mentioned permissions, that the "rw" permission remains the last entry.

This setting can not be deactivated.

Default value:

```
$Self->{'System::Customer::Permission'} =  [
  'ro',
  'rw'
];
```

LanguageDebug

Debugs the translation set. If this is set to "Yes" all strings (text) without translations are written to STDERR. This can be helpful when you are creating a new translation file. Otherwise, this option should remain set to "No".

This setting can not be deactivated.

Default value:

```
$Self->{'LanguageDebug'} =  '0';
```

Secure::DisableBanner

If enabled, the OTRS version tag will be removed from the Webinterface, the HTTP headers and the X-Headers of outgoing mails.

This setting can not be deactivated.

Default value:

```
$Self->{'Secure::DisableBanner'} =  '0';
```

Framework → Core::Cache

Cache::Module

Selects the cache backend to use.

This setting can not be deactivated.

Default value:

```
$Self->{'Cache::Module'} =  'Kernel::System::Cache::FileStorable';
```

Cache::InMemory
Should the cache data be held in memory?

This setting can not be deactivated.

Default value:

```
$Self->{'Cache::InMemory'} =  '1';
```

Cache::InBackend
Should the cache data be stored in the selected cache backend?

This setting can not be deactivated.

Default value:

```
$Self->{'Cache::InBackend'} =  '1';
```

Cache::SubdirLevels
Specify how many sub directory levels to use when creating cache files. This should prevent too many cache files being in one directory.

This setting can not be deactivated.

Default value:

```
$Self->{'Cache::SubdirLevels'} =  '2';
```

Framework → Core::CustomerCompany

CustomerCompany::EventModulePost###100-UpdateCustomerUsers
Event module that updates customer users after an update of the Customer.

Default value:

```
$Self->{'CustomerCompany::EventModulePost'}->{'100-UpdateCustomerUsers'} =  {
  'Event' => 'CustomerCompanyUpdate',
  'Module' => 'Kernel::System::CustomerCompany::Event::CustomerUserUpdate',
  'Transaction' => '0'
};
```

Framework → Core::CustomerUser

CustomerUser::EventModulePost###100-UpdateSearchProfiles
Event module that updates customer user search profiles if login changes.

Default value:

```
$Self->{'CustomerUser::EventModulePost'}->{'100-UpdateSearchProfiles'} =  {
  'Event' => 'CustomerUserUpdate',
  'Module' => 'Kernel::System::CustomerUser::Event::SearchProfileUpdate',
  'Transaction' => '0'
};
```

CustomerUser::EventModulePost###100-UpdateServiceMembership
Event module that updates customer user service membership if login changes.

Default value:

```
$Self->{'CustomerUser::EventModulePost'}->{'100-UpdateServiceMembership'} =  {
  'Event' => 'CustomerUserUpdate',
  'Module' => 'Kernel::System::CustomerUser::Event::ServiceMemberUpdate',
  'Transaction' => '0'
```

```
};
```

Framework → Core::Fetchmail

Fetchmail::Bin

Defines the fall-back path to open fetchmail binary. Note: The name of the binary needs to be 'fetchmail', if it is different please use a symbolic link.

This setting is not active by default.

Default value:

```
$Self->{'Fetchmail::Bin'} =  '/usr/bin/fetchmail';
```

Framework → Core::LinkObject

LinkObject::ViewMode

Determines the way the linked objects are displayed in each zoom mask.

This setting can not be deactivated.

Default value:

```
$Self->{'LinkObject::ViewMode'} =  'Simple';
```

LinkObject::Type###Normal

Defines the link type 'Normal'. If the source name and the target name contain the same value, the resulting link is a non-directional one; otherwise, the result is a directional link.

This setting can not be deactivated.

Default value:

```
$Self->{'LinkObject::Type'}->{'Normal'} =  {
  'SourceName' => 'Normal',
  'TargetName' => 'Normal'
};
```

LinkObject::Type###ParentChild

Defines the link type 'ParentChild'. If the source name and the target name contain the same value, the resulting link is a non-directional one; otherwise, the result is a directional link.

This setting can not be deactivated.

Default value:

```
$Self->{'LinkObject::Type'}->{'ParentChild'} =  {
  'SourceName' => 'Parent',
  'TargetName' => 'Child'
};
```

LinkObject::TypeGroup###0001

Defines the link type groups. The link types of the same group cancel one another. Example: If ticket A is linked per a 'Normal' link with ticket B, then these tickets could not be additionally linked with link of a 'ParentChild' relationship.

Default value:

```
$Self->{'LinkObject::TypeGroup'}->{'0001'} =  [
  'Normal',
  'ParentChild'
];
```

Framework → Core::Log

LogModule

Defines the log module for the system. "File" writes all messages in a given logfile, "SysLog" uses the syslog daemon of the system, e.g. syslogd.

This setting can not be deactivated.

Default value:

```
$Self->{'LogModule'} =  'Kernel::System::Log::SysLog';
```

LogModule::SysLog::Facility

If "SysLog" was selected for LogModule, a special log facility can be specified.

This setting can not be deactivated.

Default value:

```
$Self->{'LogModule::SysLog::Facility'} =  'user';
```

LogModule::SysLog::LogSock

If "SysLog" was selected for LogModule, a special log sock can be specified (on solaris you may need to use 'stream').

This setting can not be deactivated.

Default value:

```
$Self->{'LogModule::SysLog::LogSock'} =  'unix';
```

LogModule::SysLog::Charset

If "SysLog" was selected for LogModule, the charset that should be used for logging can be specified.

This setting can not be deactivated.

Default value:

```
$Self->{'LogModule::SysLog::Charset'} =  'utf-8';
```

LogModule::LogFile

If "file" was selected for LogModule, a logfile must be specified. If the file doesn't exist, it will be created by the system.

This setting can not be deactivated.

Default value:

```
$Self->{'LogModule::LogFile'} =  '/tmp/otrs.log';
```

LogModule::LogFile::Date

Adds a suffix with the actual year and month to the OTRS log file. A logfile for every month will be created.

This setting can not be deactivated.

Default value:

```
$Self->{'LogModule::LogFile::Date'} =  '0';
```

MinimumLogLevel

Set minimum loglevel. If you select 'error', just errors are logged. With 'debug' you get all logging messages.

This setting can not be deactivated.

Default value:

```
$Self->{'MinimumLogLevel'} =  'error';
```

Framework → Core::MIME-Viewer

MIME-Viewer###application/excel
Specifies the path to the converter that allows the view of Microsoft Excel files, in the web interface.

This setting is not active by default.

Default value:

```
$Self->{'MIME-Viewer'}->{'application/excel'} =  'xlhtml';
```

MIME-Viewer###application/msword
Specifies the path to the converter that allows the view of Microsoft Word files, in the web interface.

This setting is not active by default.

Default value:

```
$Self->{'MIME-Viewer'}->{'application/msword'} =  'wvWare';
```

MIME-Viewer###application/pdf
Specifies the path to the converter that allows the view of PDF documents, in the web interface.

This setting is not active by default.

Default value:

```
$Self->{'MIME-Viewer'}->{'application/pdf'} =  'pdftohtml -stdout -i';
```

MIME-Viewer###text/xml
Specifies the path to the converter that allows the view of XML files, in the web interface.

This setting is not active by default.

Default value:

```
$Self->{'MIME-Viewer'}->{'text/xml'} =  '<OTRS_CONFIG_Home>/scripts/tools/xml2html.pl';
```

Framework → Core::MirrorDB

Core::MirrorDB::DSN
OTRS can use one or more readonly mirror databases for expensive operations like fulltext search or statistics generation. Here you can specify the DSN for the first mirror database.

This setting is not active by default.

Default value:

```
$Self->{'Core::MirrorDB::DSN'} =  'DBI:mysql:database=mirrordb;host=mirrordbhost';
```

Core::MirrorDB::User
Specify the username to authenticate for the first mirror database.

This setting is not active by default.

Default value:

```
$Self->{'Core::MirrorDB::User'} =  'some_user';
```

Core::MirrorDB::Password

Specify the password to authenticate for the first mirror database.

This setting is not active by default.

Default value:

```
$Self->{'Core::MirrorDB::Password'} =  'some_password';
```

Core::MirrorDB::AdditionalMirrors###1

Configure any additional readonly mirror databases that you want to use.

This setting is not active by default.

Default value:

```
$Self->{'Core::MirrorDB::AdditionalMirrors'}->{'1'} =  {
  'DSN' => 'DBI:mysql:database=mirrordb;host=mirrordbhost',
  'Password' => 'some_password',
  'User' => 'some_user'
};
```

Core::MirrorDB::AdditionalMirrors###2

Configure any additional readonly mirror databases that you want to use.

This setting is not active by default.

Default value:

```
$Self->{'Core::MirrorDB::AdditionalMirrors'}->{'2'} =  {
  'DSN' => 'DBI:mysql:database=mirrordb;host=mirrordbhost',
  'Password' => 'some_password',
  'User' => 'some_user'
};
```

Core::MirrorDB::AdditionalMirrors###3

Configure any additional readonly mirror databases that you want to use.

This setting is not active by default.

Default value:

```
$Self->{'Core::MirrorDB::AdditionalMirrors'}->{'3'} =  {
  'DSN' => 'DBI:mysql:database=mirrordb;host=mirrordbhost',
  'Password' => 'some_password',
  'User' => 'some_user'
};
```

Core::MirrorDB::AdditionalMirrors###4

Configure any additional readonly mirror databases that you want to use.

This setting is not active by default.

Default value:

```
$Self->{'Core::MirrorDB::AdditionalMirrors'}->{'4'} =  {
  'DSN' => 'DBI:mysql:database=mirrordb;host=mirrordbhost',
  'Password' => 'some_password',
  'User' => 'some_user'
};
```

Core::MirrorDB::AdditionalMirrors###5

Configure any additional readonly mirror databases that you want to use.

This setting is not active by default.

Default value:

```
$Self->{'Core::MirrorDB::AdditionalMirrors'}->{'5'} =  {
  'DSN' => 'DBI:mysql:database=mirrordb;host=mirrordbhost',
  'Password' => 'some_password',
  'User' => 'some_user'
};
```

Framework → Core::OTRSBusiness

OTRSBusiness::ReleaseChannel

Specify the channel to be used to fetch OTRS Business Solution™ updates. Warning: Development releases might not be complete, your system might experience unrecoverable errors and on extreme cases could become unresponsive!

This setting can not be deactivated.

Default value:

```
$Self->{'OTRSBusiness::ReleaseChannel'} =  '1';
```

Framework → Core::PDF

PDF::LogoFile

Specifies the path of the file for the logo in the page header (gif|jpg|png, 700 x 100 pixel).

This setting can not be deactivated.

Default value:

```
$Self->{'PDF::LogoFile'} =  '<OTRS_CONFIG_Home>/var/logo-otrs.png';
```

PDF::PageSize

Defines the standard size of PDF pages.

This setting can not be deactivated.

Default value:

```
$Self->{'PDF::PageSize'} =  'a4';
```

PDF::MaxPages

Defines the maximum number of pages per PDF file.

This setting can not be deactivated.

Default value:

```
$Self->{'PDF::MaxPages'} =  '100';
```

PDF::TTFontFile###Proportional

Defines the path and TTF-File to handle proportional font in PDF documents.

This setting can not be deactivated.

Default value:

```
$Self->{'PDF::TTFontFile'}->{'Proportional'} =  'DejaVuSans.ttf';
```

PDF::TTFontFile###ProportionalBold

Defines the path and TTF-File to handle bold proportional font in PDF documents.

This setting can not be deactivated.

Default value:

```
$Self->{'PDF::TTFontFile'}->{'ProportionalBold'} =  'DejaVuSans-Bold.ttf';
```

PDF::TTFontFile###ProportionalItalic
Defines the path and TTF-File to handle italic proportional font in PDF documents.

This setting can not be deactivated.

Default value:

```
$Self->{'PDF::TTFontFile'}->{'ProportionalItalic'} =  'DejaVuSans-Oblique.ttf';
```

PDF::TTFontFile###ProportionalBoldItalic
Defines the path and TTF-File to handle bold italic proportional font in PDF documents.

This setting can not be deactivated.

Default value:

```
$Self->{'PDF::TTFontFile'}->{'ProportionalBoldItalic'} =  'DejaVuSans-BoldOblique.ttf';
```

PDF::TTFontFile###Monospaced
Defines the path and TTF-File to handle monospaced font in PDF documents.

This setting can not be deactivated.

Default value:

```
$Self->{'PDF::TTFontFile'}->{'Monospaced'} =  'DejaVuSansMono.ttf';
```

PDF::TTFontFile###MonospacedBold
Defines the path and TTF-File to handle bold monospaced font in PDF documents.

This setting can not be deactivated.

Default value:

```
$Self->{'PDF::TTFontFile'}->{'MonospacedBold'} =  'DejaVuSansMono-Bold.ttf';
```

PDF::TTFontFile###MonospacedItalic
Defines the path and TTF-File to handle italic monospaced font in PDF documents.

This setting can not be deactivated.

Default value:

```
$Self->{'PDF::TTFontFile'}->{'MonospacedItalic'} =  'DejaVuSansMono-Oblique.ttf';
```

PDF::TTFontFile###MonospacedBoldItalic
Defines the path and TTF-File to handle bold italic monospaced font in PDF documents.

This setting can not be deactivated.

Default value:

```
$Self->{'PDF::TTFontFile'}->{'MonospacedBoldItalic'} =  'DejaVuSansMono-
BoldOblique.ttf';
```

Framework → Core::Package

Package::FileUpload
Enables file upload in the package manager frontend.

This setting can not be deactivated.

Default value:

```
$Self->{'Package::FileUpload'} =  '1';
```

Package::RepositoryRoot

Defines the location to get online repository list for additional packages. The first available result will be used.

Default value:

```
$Self->{'Package::RepositoryRoot'} =  [
  'http://ftp.otrs.org/pub/otrs/misc/packages/repository.xml'
];
```

Package::RepositoryList

Defines the list of online repositories. Another installations can be used as repository, for example: Key="http://example.com/otrs/public.pl?Action=PublicRepository;File=" and Content="Some Name".

This setting is not active by default.

Default value:

```
$Self->{'Package::RepositoryList'} =  {
  'ftp://ftp.example.com/pub/otrs/misc/packages/' => '[Example] ftp://ftp.example.com/'
};
```

Package::RepositoryAccessRegExp

Defines the IP regular expression for accessing the local repository. You need to enable this to have access to your local repository and the package::RepositoryList is required on the remote host.

This setting is not active by default.

Default value:

```
$Self->{'Package::RepositoryAccessRegExp'} =  '127\\.0\\.0\\.1';
```

Package::Timeout

Sets the timeout (in seconds) for package downloads. Overwrites "WebUserAgent::Timeout".

This setting can not be deactivated.

Default value:

```
$Self->{'Package::Timeout'} =  '120';
```

Package::Proxy

Fetches packages via proxy. Overwrites "WebUserAgent::Proxy".

This setting is not active by default.

Default value:

```
$Self->{'Package::Proxy'} =  'http://proxy.sn.no:8001/';
```

Package::AllowLocalModifications

If this setting is active, local modifications will not be highlighted as errors in the package manager and support data collector.

This setting is not active by default.

Default value:

```
$Self->{'Package::AllowLocalModifications'} =  '0';
```

Package::ShowFeatureAddons
Toggles display of OTRS FeatureAddons list in PackageManager.

Default value:

```
$Self->{'Package::ShowFeatureAddons'} =  '1';
```

Package::EventModulePost###99-SupportDataSend
Package event module file a scheduler task for update registration.

Default value:

```
$Self->{'Package::EventModulePost'}->{'99-SupportDataSend'} =  {
  'Event' => '(PackageInstall|PackageReinstall|PackageUpgrade|PackageUninstall)',
  'Module' => 'Kernel::System::Package::Event::SupportDataSend',
  'Transaction' => '1'
};
```

Framework → Core::PerformanceLog

PerformanceLog
Enables performance log (to log the page response time). It will affect the system performance. Frontend::Module###AdminPerformanceLog must be enabled.

Default value:

```
$Self->{'PerformanceLog'} =  '0';
```

PerformanceLog::File
Specifies the path of the file for the performance log.

This setting can not be deactivated.

Default value:

```
$Self->{'PerformanceLog::File'} =  '<OTRS_CONFIG_Home>/var/log/Performance.log';
```

PerformanceLog::FileMax
Defines the maximum size (in MB) of the log file.

This setting can not be deactivated.

Default value:

```
$Self->{'PerformanceLog::FileMax'} =  '25';
```

Framework → Core::ReferenceData

ReferenceData::OwnCountryList
This setting allows you to override the built-in country list with your own list of countries. This is particularly handy if you just want to use a small select group of countries.

This setting is not active by default.

Default value:

```
$Self->{'ReferenceData::OwnCountryList'} =  {
  'AT' => 'Austria',
  'CH' => 'Switzerland',
  'DE' => 'Germany'
};
```

Framework → Core::SOAP

SOAP::User
Defines the username to access the SOAP handle (bin/cgi-bin/rpc.pl).

This setting is not active by default.

Default value:

```
$Self->{'SOAP::User'} =  'some_user';
```

SOAP::Password
Defines the password to access the SOAP handle (bin/cgi-bin/rpc.pl).

This setting is not active by default.

Default value:

```
$Self->{'SOAP::Password'} =  'some_pass';
```

SOAP::Keep-Alive
Enable keep-alive connection header for SOAP responses.

This setting can not be deactivated.

Default value:

```
$Self->{'SOAP::Keep-Alive'} =  '0';
```

Framework → Core::Sendmail

SendmailModule
Defines the module to send emails. "Sendmail" directly uses the sendmail binary of your operating system. Any of the "SMTP" mechanisms use a specified (external) mailserver. "DoNotSendEmail" doesn't send emails and it is useful for test systems.

This setting can not be deactivated.

Default value:

```
$Self->{'SendmailModule'} =  'Kernel::System::Email::Sendmail';
```

SendmailModule::CMD
If "Sendmail" was selected as SendmailModule, the location of the sendmail binary and the needed options must be specified.

This setting can not be deactivated.

Default value:

```
$Self->{'SendmailModule::CMD'} =  '/usr/sbin/sendmail -i -f';
```

SendmailModule::Host
If any of the "SMTP" mechanisms was selected as SendmailModule, the mailhost that sends out the mails must be specified.

This setting can not be deactivated.

Default value:

```
$Self->{'SendmailModule::Host'} =  'mail.example.com';
```

SendmailModule::Port
If any of the "SMTP" mechanisms was selected as SendmailModule, the port where your mailserver is listening for incoming connections must be specified.

This setting is not active by default.

Default value:

```
$Self->{'SendmailModule::Port'} =  '25';
```

SendmailModule::AuthUser
If any of the "SMTP" mechanisms was selected as SendmailModule, and authentication to the mail server is needed, an username must be specified.

This setting is not active by default.

Default value:

```
$Self->{'SendmailModule::AuthUser'} =  'MailserverLogin';
```

SendmailModule::AuthPassword
If any of the "SMTP" mechanisms was selected as SendmailModule, and authentication to the mail server is needed, a password must be specified.

This setting is not active by default.

Default value:

```
$Self->{'SendmailModule::AuthPassword'} =  'MailserverPassword';
```

SendmailBcc
Sends all outgoing email via bcc to the specified address. Please use this only for backup reasons.

Default value:

```
$Self->{'SendmailBcc'} =  '';
```

SendmailEnvelopeFrom
If set, this address is used as envelope sender in outgoing messages (not notifications - see below). If no address is specified, the envelope sender is equal to queue e-mail address.

This setting is not active by default.

Default value:

```
$Self->{'SendmailEnvelopeFrom'} =  '';
```

SendmailNotificationEnvelopeFrom
If set, this address is used as envelope sender header in outgoing notifications. If no address is specified, the envelope sender header is empty.

This setting is not active by default.

Default value:

```
$Self->{'SendmailNotificationEnvelopeFrom'} =  '';
```

SendmailEncodingForce
Forces encoding of outgoing emails (7bit|8bit|quoted-printable|base64).

This setting is not active by default.

Default value:

```
$Self->{'SendmailEncodingForce'} =  'base64';
```

Framework → Core::Session

SessionModule
Defines the module used to store the session data. With "DB" the frontend server can be splitted from the db server. "FS" is faster.

This setting can not be deactivated.

Default value:

```
$Self->{'SessionModule'} =  'Kernel::System::AuthSession::DB';
```

SessionName
Defines the name of the session key. E.g. Session, SessionID or OTRS.

This setting can not be deactivated.

Default value:

```
$Self->{'SessionName'} =  'OTRSAgentInterface';
```

CustomerPanelSessionName
Defines the name of the key for customer sessions.

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerPanelSessionName'} =  'OTRSCustomerInterface';
```

SessionCheckRemoteIP
Turns on the remote ip address check. It should be set to "No" if the application is used, for example, via a proxy farm or a dialup connection, because the remote ip address is mostly different for the requests.

This setting can not be deactivated.

Default value:

```
$Self->{'SessionCheckRemoteIP'} =  '1';
```

SessionDeleteIfNotRemoteID
Deletes a session if the session id is used with an invalid remote IP address.

This setting can not be deactivated.

Default value:

```
$Self->{'SessionDeleteIfNotRemoteID'} =  '1';
```

SessionMaxTime
Defines the maximal valid time (in seconds) for a session id.

This setting can not be deactivated.

Default value:

```
$Self->{'SessionMaxTime'} =  '57600';
```

SessionMaxIdleTime
Sets the inactivity time (in seconds) to pass before a session is killed and a user is logged out.

This setting can not be deactivated.

Default value:

```
$Self->{'SessionMaxIdleTime'} =  '21600';
```

SessionActiveTime
Sets the time (in seconds) a user is marked as active (minimum active time is 300 seconds).

This setting can not be deactivated.

Default value:

```
$Self->{'SessionActiveTime'} =  '600';
```

SessionDeleteIfTimeToOld
Deletes requested sessions if they have timed out.

This setting can not be deactivated.

Default value:

```
$Self->{'SessionDeleteIfTimeToOld'} =  '1';
```

SessionUseCookie
Makes the session management use html cookies. If html cookies are disabled or if the client browser disabled html cookies, then the system will work as usual and append the session id to the links.

This setting can not be deactivated.

Default value:

```
$Self->{'SessionUseCookie'} =  '1';
```

SessionUseCookieAfterBrowserClose
Stores cookies after the browser has been closed.

This setting can not be deactivated.

Default value:

```
$Self->{'SessionUseCookieAfterBrowserClose'} =  '0';
```

SessionCSRFProtection
Protection against CSRF (Cross Site Request Forgery) exploits (for more info see http://en.wikipedia.org/wiki/Cross-site_request_forgery).

This setting can not be deactivated.

Default value:

```
$Self->{'SessionCSRFProtection'} =  '1';
```

AgentSessionLimitPriorWarning
Sets the maximum number of active agents within the timespan defined in SessionActiveTime before a prior warning will be visible for the logged in agents.

This setting is not active by default.

Default value:

```
$Self->{'AgentSessionLimitPriorWarning'} =  '90';
```

AgentSessionLimit
Sets the maximum number of active agents within the timespan defined in SessionActiveTime.

Default value:

```
$Self->{'AgentSessionLimit'} =  '100';
```

AgentSessionPerUserLimit
Sets the maximum number of active sessions per agent within the timespan defined in SessionActiveTime.

Default value:

```
$Self->{'AgentSessionPerUserLimit'} =  '20';
```

CustomerSessionLimit
Sets the maximum number of active customers within the timespan defined in SessionActiveTime.

Default value:

```
$Self->{'CustomerSessionLimit'} =  '100';
```

CustomerSessionPerUserLimit
Sets the maximum number of active sessions per customers within the timespan defined in SessionActiveTime.

Default value:

```
$Self->{'CustomerSessionPerUserLimit'} =  '20';
```

SessionDir
If "FS" was selected for SessionModule, a directory where the session data will be stored must be specified.

This setting can not be deactivated.

Default value:

```
$Self->{'SessionDir'} =  '<OTRS_CONFIG_Home>/var/sessions';
```

SessionTable
If "DB" was selected for SessionModule, a table in database where session data will be stored must be specified.

This setting can not be deactivated.

Default value:

```
$Self->{'SessionTable'} =  'sessions';
```

Framework → Core::SpellChecker

SpellChecker
Enables spell checker support.

This setting can not be deactivated.

Default value:

```
$Self->{'SpellChecker'} =  '0';
```

SpellCheckerBin
Install ispell or aspell on the system, if you want to use a spell checker. Please specify the path to the aspell or ispell binary on your operating system.

This setting can not be deactivated.

Default value:

```
$Self->{'SpellCheckerBin'} =  '/usr/bin/ispell';
```

SpellCheckerDictDefault
Defines the default spell checker dictionary.

This setting can not be deactivated.

Default value:

```
$Self->{'SpellCheckerDictDefault'} =  'english';
```

SpellCheckerIgnore
Defines a default list of words, that are ignored by the spell checker.

This setting can not be deactivated.

Default value:

```
$Self->{'SpellCheckerIgnore'} =  [
  'www',
  'webmail',
  'https',
  'http',
  'html',
  'rfc'
];
```

Framework → Core::Stats

Stats::StatsHook
Sets the stats hook.

This setting can not be deactivated.

Default value:

```
$Self->{'Stats::StatsHook'} =  'Stat#';
```

Stats::StatsStartNumber
Start number for statistics counting. Every new stat increments this number.

This setting can not be deactivated.

Default value:

```
$Self->{'Stats::StatsStartNumber'} =  '10000';
```

Stats::MaxXaxisAttributes
Defines the default maximum number of X-axis attributes for the time scale.

This setting is not active by default.

Default value:

```
$Self->{'Stats::MaxXaxisAttributes'} =  '1000';
```

Framework → Core::Time

TimeInputFormat
Defines the date input format used in forms (option or input fields).

This setting can not be deactivated.

Default value:

```
$Self->{'TimeInputFormat'} =  'Option';
```

TimeShowAlwaysLong
Shows time in long format (days, hours, minutes), if set to "Yes"; or in short format (days, hours), if set to "No".

This setting can not be deactivated.

Default value:

```
$Self->{'TimeShowAlwaysLong'} =  '0';
```

TimeZone

Sets the system time zone (required a system with UTC as system time). Otherwise this is a diff time to the local time.

This setting is not active by default.

Default value:

```
$Self->{'TimeZone'} =  '+0';
```

TimeZoneUser

Sets the user time zone per user (required a system with UTC as system time and UTC under TimeZone). Otherwise this is a diff time to the local time.

Default value:

```
$Self->{'TimeZoneUser'} =  '0';
```

TimeZoneUserBrowserAutoOffset

Sets the user time zone per user based on java script / browser time zone offset feature at login time.

Default value:

```
$Self->{'TimeZoneUserBrowserAutoOffset'} =  '1';
```

MaximumCalendarNumber

Maximum Number of a calendar shown in a dropdown.

This setting is not active by default.

Default value:

```
$Self->{'MaximumCalendarNumber'} =  '50';
```

CalendarWeekDayStart

Define the start day of the week for the date picker.

This setting can not be deactivated.

Default value:

```
$Self->{'CalendarWeekDayStart'} =  '1';
```

TimeVacationDays

Adds the permanent vacation days. Please use single digit pattern for numbers from 1 to 9 (instead of 01 - 09).

This setting can not be deactivated.

Default value:

```
$Self->{'TimeVacationDays'} =  {
 '1' => {
   '1' => 'New Year\'s Day'
 },
 '12' => {
   '24' => 'Christmas Eve',
   '25' => 'First Christmas Day',
   '26' => 'Second Christmas Day',
   '31' => 'New Year\'s Eve'
 },
```

```
  '5' => {
    '1' => 'International Workers\' Day'
  }
};
```

TimeVacationDaysOneTime

Adds the one time vacation days. Please use single digit pattern for numbers from 1 to 9 (instead of 01 - 09).

This setting can not be deactivated.

Default value:

```
$Self->{'TimeVacationDaysOneTime'} =  {
  '2004' => {
    '1' => {
      '1' => 'test'
    }
  }
};
```

TimeWorkingHours

Defines the hours and week days to count the working time.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeWorkingHours'} =  {
  'Fri' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Mon' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Sat' => [],
  'Sun' => [],
  'Thu' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
```

```
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Tue' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Wed' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ]
};
```

TimeShowCompleteDescription

Shows time use complete description (days, hours, minutes), if set to "Yes"; or just first letter (d, h, m), if set to "No".

This setting can not be deactivated.

Default value:

```
$Self->{'TimeShowCompleteDescription'} =  '0';
```

Framework → Core::Time::Calendar1

TimeZone::Calendar1Name

Defines the name of the indicated calendar.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeZone::Calendar1Name'} =  'Calendar Name 1';
```

TimeZone::Calendar1

Defines the time zone of the indicated calendar, which can be assigned later to a specific queue.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeZone::Calendar1'} =  '0';
```

CalendarWeekDayStart::Calendar1

Define the start day of the week for the date picker for the indicated calendar.

This setting can not be deactivated.

Default value:

```
$Self->{'CalendarWeekDayStart::Calendar1'} =  '1';
```

TimeVacationDays::Calendar1

Adds the permanent vacation days for the indicated calendar. Please use single digit pattern for numbers from 1 to 9 (instead of 01 - 09).

This setting can not be deactivated.

Default value:

```
$Self->{'TimeVacationDays::Calendar1'} =  {
 '1' => {
   '1' => 'New Year\'s Day'
 },
 '12' => {
   '24' => 'Christmas Eve',
   '25' => 'First Christmas Day',
   '26' => 'Second Christmas Day',
   '31' => 'New Year\'s Eve'
 },
 '5' => {
   '1' => 'International Workers\' Day'
 }
};
```

TimeVacationDaysOneTime::Calendar1

Adds the one time vacation days for the indicated calendar. Please use single digit pattern for numbers from 1 to 9 (instead of 01 - 09).

This setting can not be deactivated.

Default value:

```
$Self->{'TimeVacationDaysOneTime::Calendar1'} =  {
 '2004' => {
   '1' => {
     '1' => 'test'
   }
 }
};
```

TimeWorkingHours::Calendar1

Defines the hours and week days of the indicated calendar, to count the working time.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeWorkingHours::Calendar1'} =  {
 'Fri' => [
   '8',
   '9',
   '10',
   '11',
   '12',
   '13',
   '14',
   '15',
   '16',
   '17',
   '18',
```

```
    '19',
    '20'
  ],
  'Mon' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Sat' => [],
  'Sun' => [],
  'Thu' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Tue' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Wed' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ]
};
```

Framework → Core::Time::Calendar2

TimeZone::Calendar2Name

Defines the name of the indicated calendar.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeZone::Calendar2Name'} =  'Calendar Name 2';
```

TimeZone::Calendar2

Defines the time zone of the indicated calendar, which can be assigned later to a specific queue.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeZone::Calendar2'} =  '0';
```

CalendarWeekDayStart::Calendar2

Define the start day of the week for the date picker for the indicated calendar.

This setting can not be deactivated.

Default value:

```
$Self->{'CalendarWeekDayStart::Calendar2'} =  '1';
```

TimeVacationDays::Calendar2

Adds the permanent vacation days for the indicated calendar. Please use single digit pattern for numbers from 1 to 9 (instead of 01 - 09).

This setting can not be deactivated.

Default value:

```
$Self->{'TimeVacationDays::Calendar2'} =  {
  '1' => {
    '1' => 'New Year\'s Day'
  },
  '12' => {
    '24' => 'Christmas Eve',
    '25' => 'First Christmas Day',
    '26' => 'Second Christmas Day',
    '31' => 'New Year\'s Eve'
  },
  '5' => {
    '1' => 'International Workers\' Day'
  }
};
```

TimeVacationDaysOneTime::Calendar2

Adds the one time vacation days for the indicated calendar. Please use single digit pattern for numbers from 1 to 9 (instead of 01 - 09).

This setting can not be deactivated.

Default value:

```
$Self->{'TimeVacationDaysOneTime::Calendar2'} =  {
  '2004' => {
    '1' => {
      '1' => 'test'
    }
  }
};
```

TimeWorkingHours::Calendar2

Defines the hours and week days of the indicated calendar, to count the working time.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeWorkingHours::Calendar2'} =  {
 'Fri' => [
   '8',
   '9',
   '10',
   '11',
   '12',
   '13',
   '14',
   '15',
   '16',
   '17',
   '18',
   '19',
   '20'
 ],
 'Mon' => [
   '8',
   '9',
   '10',
   '11',
   '12',
   '13',
   '14',
   '15',
   '16',
   '17',
   '18',
   '19',
   '20'
 ],
 'Sat' => [],
 'Sun' => [],
 'Thu' => [
   '8',
   '9',
   '10',
   '11',
   '12',
   '13',
   '14',
   '15',
   '16',
   '17',
   '18',
   '19',
   '20'
 ],
 'Tue' => [
   '8',
   '9',
   '10',
   '11',
   '12',
   '13',
   '14',
   '15',
   '16',
   '17',
   '18',
   '19',
   '20'
 ],
 'Wed' => [
   '8',
   '9',
   '10',
   '11',
```

```
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ]
};
```

Framework → Core::Time::Calendar3

TimeZone::Calendar3Name

Defines the name of the indicated calendar.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeZone::Calendar3Name'} =  'Calendar Name 3';
```

TimeZone::Calendar3

Defines the time zone of the indicated calendar, which can be assigned later to a specific queue.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeZone::Calendar3'} =  '0';
```

CalendarWeekDayStart::Calendar3

Define the start day of the week for the date picker for the indicated calendar.

This setting can not be deactivated.

Default value:

```
$Self->{'CalendarWeekDayStart::Calendar3'} =  '1';
```

TimeVacationDays::Calendar3

Adds the permanent vacation days for the indicated calendar. Please use single digit pattern for numbers from 1 to 9 (instead of 01 - 09).

This setting can not be deactivated.

Default value:

```
$Self->{'TimeVacationDays::Calendar3'} =  {
  '1' => {
    '1' => 'New Year\'s Day'
  },
  '12' => {
    '24' => 'Christmas Eve',
    '25' => 'First Christmas Day',
    '26' => 'Second Christmas Day',
    '31' => 'New Year\'s Eve'
  },
  '5' => {
    '1' => 'International Workers\' Day'
  }
};
```

TimeVacationDaysOneTime::Calendar3

Adds the one time vacation days for the indicated calendar. Please use single digit pattern for numbers from 1 to 9 (instead of 01 - 09).

This setting can not be deactivated.

Default value:

```
$Self->{'TimeVacationDaysOneTime::Calendar3'} =  {
 '2004' => {
   '1' => {
     '1' => 'test'
   }
 }
};
```

TimeWorkingHours::Calendar3

Defines the hours and week days of the indicated calendar, to count the working time.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeWorkingHours::Calendar3'} =  {
 'Fri' => [
   '8',
   '9',
   '10',
   '11',
   '12',
   '13',
   '14',
   '15',
   '16',
   '17',
   '18',
   '19',
   '20'
 ],
 'Mon' => [
   '8',
   '9',
   '10',
   '11',
   '12',
   '13',
   '14',
   '15',
   '16',
   '17',
   '18',
   '19',
   '20'
 ],
 'Sat' => [],
 'Sun' => [],
 'Thu' => [
   '8',
   '9',
   '10',
   '11',
   '12',
   '13',
   '14',
   '15',
   '16',
   '17',
   '18',
   '19',
   '20'
 ],
 'Tue' => [
   '8',
   '9',
   '10',
```

```
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Wed' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ]
};
```

Framework → Core::Time::Calendar4

TimeZone::Calendar4Name

Defines the name of the indicated calendar.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeZone::Calendar4Name'} =  'Calendar Name 4';
```

TimeZone::Calendar4

Defines the time zone of the indicated calendar, which can be assigned later to a specific queue.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeZone::Calendar4'} =  '0';
```

CalendarWeekDayStart::Calendar4

Define the start day of the week for the date picker for the indicated calendar.

This setting can not be deactivated.

Default value:

```
$Self->{'CalendarWeekDayStart::Calendar4'} =  '1';
```

TimeVacationDays::Calendar4

Adds the permanent vacation days for the indicated calendar. Please use single digit pattern for numbers from 1 to 9 (instead of 01 - 09).

This setting can not be deactivated.

Default value:

```
$Self->{'TimeVacationDays::Calendar4'} =  {
 '1' => {
```

```
    '1' => 'New Year\'s Day'
  },
  '12' => {
    '24' => 'Christmas Eve',
    '25' => 'First Christmas Day',
    '26' => 'Second Christmas Day',
    '31' => 'New Year\'s Eve'
  },
  '5' => {
    '1' => 'International Workers\' Day'
  }
};
```

TimeVacationDaysOneTime::Calendar4

Adds the one time vacation days for the indicated calendar. Please use single digit pattern for numbers from 1 to 9 (instead of 01 - 09).

This setting can not be deactivated.

Default value:

```
$Self->{'TimeVacationDaysOneTime::Calendar4'} =  {
  '2004' => {
    '1' => {
      '1' => 'test'
    }
  }
};
```

TimeWorkingHours::Calendar4

Defines the hours and week days of the indicated calendar, to count the working time.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeWorkingHours::Calendar4'} =  {
  'Fri' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Mon' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Sat' => [],
  'Sun' => [],
  'Thu' => [
```

```
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Tue' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Wed' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ]
};
```

Framework → Core::Time::Calendar5

TimeZone::Calendar5Name

Defines the name of the indicated calendar.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeZone::Calendar5Name'} =  'Calendar Name 5';
```

TimeZone::Calendar5

Defines the time zone of the indicated calendar, which can be assigned later to a specific queue.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeZone::Calendar5'} =  '0';
```

CalendarWeekDayStart::Calendar5

Define the start day of the week for the date picker for the indicated calendar.

This setting can not be deactivated.

Default value:

```
$Self->{'CalendarWeekDayStart::Calendar5'} =  '1';
```

TimeVacationDays::Calendar5

Adds the permanent vacation days for the indicated calendar. Please use single digit pattern for numbers from 1 to 9 (instead of 01 - 09).

This setting can not be deactivated.

Default value:

```
$Self->{'TimeVacationDays::Calendar5'} =  {
  '1' => {
    '1' => 'New Year\'s Day'
  },
  '12' => {
    '24' => 'Christmas Eve',
    '25' => 'First Christmas Day',
    '26' => 'Second Christmas Day',
    '31' => 'New Year\'s Eve'
  },
  '5' => {
    '1' => 'International Workers\' Day'
  }
};
```

TimeVacationDaysOneTime::Calendar5

Adds the one time vacation days for the indicated calendar. Please use single digit pattern for numbers from 1 to 9 (instead of 01 - 09).

This setting can not be deactivated.

Default value:

```
$Self->{'TimeVacationDaysOneTime::Calendar5'} =  {
  '2004' => {
    '1' => {
      '1' => 'test'
    }
  }
};
```

TimeWorkingHours::Calendar5

Defines the hours and week days of the indicated calendar, to count the working time.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeWorkingHours::Calendar5'} =  {
  'Fri' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
```

```
  ],
  'Mon' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Sat' => [],
  'Sun' => [],
  'Thu' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Tue' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Wed' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ]
};
```

Framework → Core::Time::Calendar6

TimeZone::Calendar6Name

Defines the name of the indicated calendar.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeZone::Calendar6Name'} =  'Calendar Name 6';
```

TimeZone::Calendar6

Defines the time zone of the indicated calendar, which can be assigned later to a specific queue.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeZone::Calendar6'} =  '0';
```

CalendarWeekDayStart::Calendar6

Define the start day of the week for the date picker for the indicated calendar.

This setting can not be deactivated.

Default value:

```
$Self->{'CalendarWeekDayStart::Calendar6'} =  '1';
```

TimeVacationDays::Calendar6

Adds the permanent vacation days for the indicated calendar. Please use single digit pattern for numbers from 1 to 9 (instead of 01 - 09).

This setting can not be deactivated.

Default value:

```
$Self->{'TimeVacationDays::Calendar6'} =  {
 '1' => {
   '1' => 'New Year\'s Day'
 },
 '12' => {
   '24' => 'Christmas Eve',
   '25' => 'First Christmas Day',
   '26' => 'Second Christmas Day',
   '31' => 'New Year\'s Eve'
 },
 '5' => {
   '1' => 'International Workers\' Day'
 }
};
```

TimeVacationDaysOneTime::Calendar6

Adds the one time vacation days for the indicated calendar. Please use single digit pattern for numbers from 1 to 9 (instead of 01 - 09).

This setting can not be deactivated.

Default value:

```
$Self->{'TimeVacationDaysOneTime::Calendar6'} =  {
 '2004' => {
   '1' => {
     '1' => 'test'
   }
 }
};
```

TimeWorkingHours::Calendar6

Defines the hours and week days of the indicated calendar, to count the working time.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeWorkingHours::Calendar6'} =  {
 'Fri' => [
   '8',
   '9',
   '10',
   '11',
   '12',
   '13',
   '14',
   '15',
   '16',
   '17',
   '18',
   '19',
   '20'
 ],
 'Mon' => [
   '8',
   '9',
   '10',
   '11',
   '12',
   '13',
   '14',
   '15',
   '16',
   '17',
   '18',
   '19',
   '20'
 ],
 'Sat' => [],
 'Sun' => [],
 'Thu' => [
   '8',
   '9',
   '10',
   '11',
   '12',
   '13',
   '14',
   '15',
   '16',
   '17',
   '18',
   '19',
   '20'
 ],
 'Tue' => [
   '8',
   '9',
   '10',
   '11',
   '12',
   '13',
   '14',
   '15',
   '16',
   '17',
   '18',
   '19',
   '20'
 ],
 'Wed' => [
   '8',
   '9',
   '10',
   '11',
```

```
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ]
};
```

Framework → Core::Time::Calendar7

TimeZone::Calendar7Name

Defines the name of the indicated calendar.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeZone::Calendar7Name'} =  'Calendar Name 7';
```

TimeZone::Calendar7

Defines the time zone of the indicated calendar, which can be assigned later to a specific queue.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeZone::Calendar7'} =  '0';
```

CalendarWeekDayStart::Calendar7

Define the start day of the week for the date picker for the indicated calendar.

This setting can not be deactivated.

Default value:

```
$Self->{'CalendarWeekDayStart::Calendar7'} =  '1';
```

TimeVacationDays::Calendar7

Adds the permanent vacation days for the indicated calendar. Please use single digit pattern for numbers from 1 to 9 (instead of 01 - 09).

This setting can not be deactivated.

Default value:

```
$Self->{'TimeVacationDays::Calendar7'} =  {
  '1' => {
    '1' => 'New Year\'s Day'
  },
  '12' => {
    '24' => 'Christmas Eve',
    '25' => 'First Christmas Day',
    '26' => 'Second Christmas Day',
    '31' => 'New Year\'s Eve'
  },
  '5' => {
    '1' => 'International Workers\' Day'
  }
};
```

TimeVacationDaysOneTime::Calendar7

Adds the one time vacation days for the indicated calendar. Please use single digit pattern for numbers from 1 to 9 (instead of 01 - 09).

This setting can not be deactivated.

Default value:

```
$Self->{'TimeVacationDaysOneTime::Calendar7'} =  {
 '2004' => {
   '1' => {
     '1' => 'test'
   }
 }
};
```

TimeWorkingHours::Calendar7

Defines the hours and week days of the indicated calendar, to count the working time.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeWorkingHours::Calendar7'} =  {
 'Fri' => [
   '8',
   '9',
   '10',
   '11',
   '12',
   '13',
   '14',
   '15',
   '16',
   '17',
   '18',
   '19',
   '20'
 ],
 'Mon' => [
   '8',
   '9',
   '10',
   '11',
   '12',
   '13',
   '14',
   '15',
   '16',
   '17',
   '18',
   '19',
   '20'
 ],
 'Sat' => [],
 'Sun' => [],
 'Thu' => [
   '8',
   '9',
   '10',
   '11',
   '12',
   '13',
   '14',
   '15',
   '16',
   '17',
   '18',
   '19',
   '20'
 ],
 'Tue' => [
   '8',
   '9',
   '10',
```

```
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Wed' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ]
};
```

Framework → Core::Time::Calendar8

TimeZone::Calendar8Name

Defines the name of the indicated calendar.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeZone::Calendar8Name'} =  'Calendar Name 8';
```

TimeZone::Calendar8

Defines the time zone of the indicated calendar, which can be assigned later to a specific queue.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeZone::Calendar8'} =  '0';
```

CalendarWeekDayStart::Calendar8

Define the start day of the week for the date picker for the indicated calendar.

This setting can not be deactivated.

Default value:

```
$Self->{'CalendarWeekDayStart::Calendar8'} =  '1';
```

TimeVacationDays::Calendar8

Adds the permanent vacation days for the indicated calendar. Please use single digit pattern for numbers from 1 to 9 (instead of 01 - 09).

This setting can not be deactivated.

Default value:

```
$Self->{'TimeVacationDays::Calendar8'} =  {
 '1' => {
```

```
    '1' => 'New Year\'s Day'
  },
  '12' => {
    '24' => 'Christmas Eve',
    '25' => 'First Christmas Day',
    '26' => 'Second Christmas Day',
    '31' => 'New Year\'s Eve'
  },
  '5' => {
    '1' => 'International Workers\' Day'
  }
};
```

TimeVacationDaysOneTime::Calendar8

Adds the one time vacation days for the indicated calendar. Please use single digit pattern for numbers from 1 to 9 (instead of 01 - 09).

This setting can not be deactivated.

Default value:

```
$Self->{'TimeVacationDaysOneTime::Calendar8'} =  {
  '2004' => {
    '1' => {
      '1' => 'test'
    }
  }
};
```

TimeWorkingHours::Calendar8

Defines the hours and week days of the indicated calendar, to count the working time.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeWorkingHours::Calendar8'} =  {
  'Fri' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Mon' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Sat' => [],
  'Sun' => [],
  'Thu' => [
```

```
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Tue' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Wed' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ]
};
```

Framework → Core::Time::Calendar9

TimeZone::Calendar9Name

Defines the name of the indicated calendar.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeZone::Calendar9Name'} =  'Calendar Name 9';
```

TimeZone::Calendar9

Defines the time zone of the indicated calendar, which can be assigned later to a specific queue.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeZone::Calendar9'} =  '0';
```

CalendarWeekDayStart::Calendar9

Define the start day of the week for the date picker for the indicated calendar.

This setting can not be deactivated.

Default value:

```
$Self->{'CalendarWeekDayStart::Calendar9'} =  '1';
```

TimeVacationDays::Calendar9

Adds the permanent vacation days for the indicated calendar. Please use single digit pattern for numbers from 1 to 9 (instead of 01 - 09).

This setting can not be deactivated.

Default value:

```
$Self->{'TimeVacationDays::Calendar9'} =  {
  '1' => {
    '1' => 'New Year\'s Day'
  },
  '12' => {
    '24' => 'Christmas Eve',
    '25' => 'First Christmas Day',
    '26' => 'Second Christmas Day',
    '31' => 'New Year\'s Eve'
  },
  '5' => {
    '1' => 'International Workers\' Day'
  }
};
```

TimeVacationDaysOneTime::Calendar9

Adds the one time vacation days for the indicated calendar. Please use single digit pattern for numbers from 1 to 9 (instead of 01 - 09).

This setting can not be deactivated.

Default value:

```
$Self->{'TimeVacationDaysOneTime::Calendar9'} =  {
  '2004' => {
    '1' => {
      '1' => 'test'
    }
  }
};
```

TimeWorkingHours::Calendar9

Defines the hours and week days of the indicated calendar, to count the working time.

This setting can not be deactivated.

Default value:

```
$Self->{'TimeWorkingHours::Calendar9'} =  {
  'Fri' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Mon' => [
```

```
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Sat' => [],
  'Sun' => [],
  'Thu' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Tue' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ],
  'Wed' => [
    '8',
    '9',
    '10',
    '11',
    '12',
    '13',
    '14',
    '15',
    '16',
    '17',
    '18',
    '19',
    '20'
  ]
};
```

Framework → Core::Web

Frontend::WebPath

Defines the URL base path of icons, CSS and Java Script.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::WebPath'} =  '/otrs-web/';
```

Frontend::ImagePath
Defines the URL image path of icons for navigation.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::ImagePath'} =  '<OTRS_CONFIG_Frontend::WebPath>skins/Agent/default/
img/';
```

Frontend::CSSPath
Defines the URL CSS path.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::CSSPath'} =  '<OTRS_CONFIG_Frontend::WebPath>css/';
```

Frontend::JavaScriptPath
Defines the URL java script path.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::JavaScriptPath'} =  '<OTRS_CONFIG_Frontend::WebPath>js/';
```

Frontend::RichText
Uses richtext for viewing and editing: articles, salutations, signatures, standard templates, auto responses and notifications.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::RichText'} =  '1';
```

Frontend::RichTextPath
Defines the URL rich text editor path.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::RichTextPath'} =  '<OTRS_CONFIG_Frontend::WebPath>js/thirdparty/
ckeditor-4.5.6/';
```

Frontend::RichTextWidth
Defines the width for the rich text editor component. Enter number (pixels) or percent value (relative).

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::RichTextWidth'} =  '620';
```

Frontend::RichTextHeight
Defines the height for the rich text editor component. Enter number (pixels) or percent value (relative).

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::RichTextHeight'} =  '320';
```

Frontend::RichText::DefaultCSS
Defines the default CSS used in rich text editors.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::RichText::DefaultCSS'} =  'font-family:Geneva,Helvetica,Arial,sans-serif; font-size: 12px;';
```

Frontend::RichText::EnhancedMode
Defines if the enhanced mode should be used (enables use of table, replace, subscript, superscript, paste from word, etc.).

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::RichText::EnhancedMode'} =  '0';
```

Frontend::RichText::EnhancedMode::Customer
Defines if the enhanced mode should be used (enables use of table, replace, subscript, superscript, paste from word, etc.) in customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::RichText::EnhancedMode::Customer'} =  '0';
```

DisableMSIFrameSecurityRestricted
Disable restricted security for IFrames in IE. May be required for SSO to work in IE.

Default value:

```
$Self->{'DisableMSIFrameSecurityRestricted'} =  '0';
```

DisableIFrameOriginRestricted
Disable HTTP header "X-Frame-Options: SAMEORIGIN" to allow OTRS to be included as an IFrame in other websites. Disabling this HTTP header can be a security issue! Only disable it, if you know what you are doing!

Default value:

```
$Self->{'DisableIFrameOriginRestricted'} =  '0';
```

DisableContentSecurityPolicy
Disable HTTP header "Content-Security-Policy" to allow loading of external script contents. Disabling this HTTP header can be a security issue! Only disable it, if you know what you are doing!

Default value:

```
$Self->{'DisableContentSecurityPolicy'} =  '0';
```

DefaultViewNewLine
Automated line break in text messages after x number of chars.

This setting can not be deactivated.

Default value:

```
$Self->{'DefaultViewNewLine'} =  '90';
```

DefaultViewLines
Sets the number of lines that are displayed in text messages (e.g. ticket lines in the QueueZoom).

This setting can not be deactivated.

Default value:

```
$Self->{'DefaultViewLines'} =  '6000';
```

Frontend::AnimationEnabled
Turns on the animations used in the GUI. If you have problems with these animations (e.g. performance issues), you can turn them off here.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::AnimationEnabled'} =  '1';
```

Frontend::MenuDragDropEnabled
Turns on drag and drop for the main navigation.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::MenuDragDropEnabled'} =  '1';
```

AttachmentDownloadType
Allows choosing between showing the attachments of a ticket in the browser (inline) or just make them downloadable (attachment).

This setting can not be deactivated.

Default value:

```
$Self->{'AttachmentDownloadType'} =  'attachment';
```

WebMaxFileUpload
Defines the maximal size (in bytes) for file uploads via the browser. Warning: Setting this option to a value which is too low could cause many masks in your OTRS instance to stop working (probably any mask which takes input from the user).

This setting can not be deactivated.

Default value:

```
$Self->{'WebMaxFileUpload'} =  '24000000';
```

WebUploadCacheModule
Selects the module to handle uploads via the web interface. "DB" stores all uploads in the database, "FS" uses the file system.

This setting can not be deactivated.

Default value:

```
$Self->{'WebUploadCacheModule'} =  'Kernel::System::Web::UploadCache::DB';
```

Frontend::Output::FilterText###AAAURL
Defines the filter that processes the text in the articles, in order to highlight URLs.

Default value:

```
$Self->{'Frontend::Output::FilterText'}->{'AAAURL'} =  {
  'Module' => 'Kernel::Output::HTML::FilterText::URL',
  'Templates' => {
    'AgentTicketZoom' => '1'
  }
};
```

Frontend::Themes

Activates the available themes on the system. Value 1 means active, 0 means inactive.

Default value:

```
$Self->{'Frontend::Themes'} =  {
  'Lite' => '0',
  'Standard' => '1'
};
```

Frontend::Output::FilterText###OutputFilterTextAutoLink

Defines a filter to process the text in the articles, in order to highlight predefined keywords.

This setting is not active by default.

Default value:

```
$Self->{'Frontend::Output::FilterText'}->{'OutputFilterTextAutoLink'} =  {
  'Module' => 'Kernel::Output::HTML::FilterText::AutoLink',
  'Templates' => {
    'AgentTicketZoom' => '1'
  }
};
```

Frontend::Output::OutputFilterTextAutoLink###CVE

Defines a filter for html output to add links behind CVE numbers. The element Image allows two input kinds. At once the name of an image (e.g. faq.png). In this case the OTRS image path will be used. The second possiblity is to insert the link to the image.

This setting is not active by default.

Default value:

```
$Self->{'Frontend::Output::OutputFilterTextAutoLink'}->{'CVE'} =  {
  'RegExp' => [
    '(CVE|CAN)\\-(\\d{3,4})\\-(\\d{2,})'
  ],
  'Templates' => {
    'AgentTicketZoom' => '1'
  },
  'URL1' => {
    'Description' => 'Mitre',
    'Image' => 'http://cve.mitre.org/favicon.ico',
    'Target' => '_blank',
    'URL' => 'http://cve.mitre.org/cgi-bin/cvename.cgi?name=<MATCH1>-<MATCH2>-<MATCH3>'
  },
  'URL2' => {
    'Description' => 'Google',
    'Image' => 'http://www.google.de/favicon.ico',
    'Target' => '_blank',
    'URL' => 'http://google.com/search?q=<MATCH1>-<MATCH2>-<MATCH3>'
  },
  'URL3' => {
    'Description' => 'US-CERT NVD',
    'Image' => 'http://nvd.nist.gov/favicon.ico',
    'Target' => '_blank',
    'URL' => 'http://nvd.nist.gov/nvd.cfm?cvename=<MATCH1>-<MATCH2>-<MATCH3>'
  }
```

```
};
```

Frontend::Output::OutputFilterTextAutoLink###Bugtraq

Defines a filter for html output to add links behind bugtraq numbers. The element Image allows two input kinds. At once the name of an image (e.g. faq.png). In this case the OTRS image path will be used. The second possiblity is to insert the link to the image.

This setting is not active by default.

Default value:

```
$Self->{'Frontend::Output::OutputFilterTextAutoLink'}->{'Bugtraq'} =  {
  'RegExp' => [
    'Bugtraq[\\s\\w\\t]*?ID[\\s\\w\\t]*?:[\\s\\w\\t]*?(\\d{2,8})',
    'Bugtraq[\\s\\w\\t]*?ID[\\s\\w\\t]*?(\\d{2,8})',
    'Bugtraq[\\s\\w\\t]*?:[\\s\\w\\t]*?(\\d{2,8})',
    'Bugtraq[\\s\\w\\t]*?(\\d{2,8})',
    'BID[\\s\\w\\t]*?:[\\s\\w\\t]*?(\\d{2,8})',
    'BID[\\s\\w\\t]*?(\\d{2,8})'
  ],
  'Templates' => {
    'AgentTicketZoom' => '1'
  },
  'URL1' => {
    'Description' => 'Security Focus',
    'Image' => 'http://www.securityfocus.com/favicon.ico',
    'Target' => '_blank',
    'URL' => 'http://www.securityfocus.com/bid/<MATCH1>/info'
  },
  'URL2' => {
    'Description' => 'Google',
    'Image' => 'http://www.google.de/favicon.ico',
    'Target' => '_blank',
    'URL' => 'http://google.com/search?q=<MATCH>'
  }
};
```

Frontend::Output::OutputFilterTextAutoLink###MSBulletins

Defines a filter for html output to add links behind MSBulletin numbers. The element Image allows two input kinds. At once the name of an image (e.g. faq.png). In this case the OTRS image path will be used. The second possiblity is to insert the link to the image.

This setting is not active by default.

Default value:

```
$Self->{'Frontend::Output::OutputFilterTextAutoLink'}->{'MSBulletins'} =  {
  'RegExp' => [
    'MS[^A-Za-z]{0,5}(\\d\\d).?(\\d{2,4})'
  ],
  'Templates' => {
    'AgentTicketZoom' => '1'
  },
  'URL1' => {
    'Description' => 'Microsoft Technet',
    'Image' => 'http://www.microsoft.com/favicon.ico',
    'Target' => '_blank',
    'URL' => 'http://www.microsoft.com/technet/security/bulletin/MS<MATCH1>-
<MATCH2>.mspx'
  },
  'URL2' => {
    'Description' => 'Google',
    'Image' => 'http://www.google.de/favicon.ico',
    'Target' => '_blank',
    'URL' => 'http://google.com/search?q=MS<MATCH1>-<MATCH2>'
  }
};
```

Frontend::Output::OutputFilterTextAutoLink###Setting1

Define a filter for html output to add links behind a defined string. The element Image allows two input kinds. At once the name of an image (e.g. faq.png). In this case the OTRS image path will be used. The second possiblity is to insert the link to the image.

This setting is not active by default.

Default value:

```
$Self->{'Frontend::Output::OutputFilterTextAutoLink'}->{'Setting1'} =  {
  'RegExp' => [
    'RegExp'
  ],
  'Templates' => {
    'AgentTicketZoom' => '1'
  },
  'URL1' => {
    'Description' => 'Description',
    'Image' => 'right-small.png',
    'Target' => '_blank',
    'URL' => 'URL'
  },
  'URL2' => {
    'Description' => 'Description',
    'Image' => 'Image',
    'Target' => '_blank',
    'URL' => 'URL'
  }
};
```

Frontend::Output::OutputFilterTextAutoLink###Setting2

Defines a filter for html output to add links behind a defined string. The element Image allows two input kinds. At once the name of an image (e.g. faq.png). In this case the OTRS image path will be used. The second possiblity is to insert the link to the image.

This setting is not active by default.

Default value:

```
$Self->{'Frontend::Output::OutputFilterTextAutoLink'}->{'Setting2'} =  {
  'RegExp' => [
    'RegExp'
  ],
  'Templates' => {
    'AgentTicketZoom' => '1'
  },
  'URL1' => {
    'Description' => 'Description',
    'Image' => 'right-small.png',
    'Target' => '_blank',
    'URL' => 'URL'
  },
  'URL2' => {
    'Description' => 'Description',
    'Image' => 'Image',
    'Target' => '_blank',
    'URL' => 'URL'
  },
  'URL3' => {
    'Description' => 'Description',
    'Image' => 'Image',
    'Target' => '_blank',
    'URL' => 'URL'
  }
};
```

Loader::Enabled::CSS

If enabled, OTRS will deliver all CSS files in minified form. WARNING: If you turn this off, there will likely be problems in IE 7, because it cannot load more than 32 CSS files.

This setting can not be deactivated.

Default value:

```
$Self->{'Loader::Enabled::CSS'} =  '1';
```

Loader::Enabled::JS
If enabled, OTRS will deliver all JavaScript files in minified form.

This setting can not be deactivated.

Default value:

```
$Self->{'Loader::Enabled::JS'} =  '1';
```

Loader::Agent::CommonCSS###000-Framework
List of CSS files to always be loaded for the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Loader::Agent::CommonCSS'}->{'000-Framework'} =  [
  'Core.Reset.css',
  'Core.Default.css',
  'Core.Header.css',
  'Core.OverviewControl.css',
  'Core.OverviewSmall.css',
  'Core.OverviewMedium.css',
  'Core.OverviewLarge.css',
  'Core.Footer.css',
  'Core.PageLayout.css',
  'Core.Form.css',
  'Core.Table.css',
  'Core.Widget.css',
  'Core.WidgetMenu.css',
  'Core.TicketDetail.css',
  'Core.Tooltip.css',
  'Core.Dialog.css',
  'Core.InputFields.css',
  'Core.Print.css',
  'thirdparty/fontawesome/font-awesome.css'
];
```

Loader::Agent::ResponsiveCSS###000-Framework
List of responsive CSS files to always be loaded for the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Loader::Agent::ResponsiveCSS'}->{'000-Framework'} =  [
  'Core.Responsive.css'
];
```

Loader::Agent::CommonJS###000-Framework
List of JS files to always be loaded for the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Loader::Agent::CommonJS'}->{'000-Framework'} =  [
  'thirdparty/jquery-2.1.4/jquery.js',
  'thirdparty/jquery-browser-detection/jquery-browser-detection.js',
  'thirdparty/jquery-ui-1.11.4/jquery-ui.js',
  'thirdparty/jquery-ui-touch-punch-0.2.3/jquery.ui.touch-punch.js',
```

```
  'thirdparty/jquery-validate-1.14.0/jquery.validate.js',
  'thirdparty/stacktrace-0.6.4/stacktrace.js',
  'thirdparty/jquery-pubsub/pubsub.js',
  'thirdparty/jquery-jstree-3.1.1/jquery.jstree.js',
  'Core.JavaScriptEnhancements.js',
  'Core.Debug.js',
  'Core.Exception.js',
  'Core.Data.js',
  'Core.Config.js',
  'Core.JSON.js',
  'Core.App.js',
  'Core.App.Responsive.js',
  'Core.AJAX.js',
  'Core.UI.js',
  'Core.UI.InputFields.js',
  'Core.UI.Accordion.js',
  'Core.UI.Datepicker.js',
  'Core.UI.DnD.js',
  'Core.UI.Floater.js',
  'Core.UI.Resizable.js',
  'Core.UI.Table.js',
  'Core.UI.Accessibility.js',
  'Core.UI.RichTextEditor.js',
  'Core.UI.Dialog.js',
  'Core.UI.ActionRow.js',
  'Core.UI.Popup.js',
  'Core.UI.TreeSelection.js',
  'Core.UI.Autocomplete.js',
  'Core.Form.js',
  'Core.Form.ErrorTooltips.js',
  'Core.Form.Validate.js',
  'Core.Agent.js',
  'Core.Agent.Search.js',
  'Core.Agent.CustomerInformationCenterSearch.js',
  'Core.UI.Notification.js',
  'Core.Agent.Responsive.js'
];
```

Loader::Agent::CommonJS###001-JQueryMigrate

List of JS files to always be loaded for the agent interface.

This setting is not active by default.

Default value:

```
 $Self->{'Loader::Agent::CommonJS'}->{'001-JQueryMigrate'} =  [
  'thirdparty/jquery-migrate-1.2.1/jquery-migrate.js'
];
```

Loader::Customer::CommonCSS###000-Framework

List of CSS files to always be loaded for the customer interface.

This setting can not be deactivated.

Default value:

```
 $Self->{'Loader::Customer::CommonCSS'}->{'000-Framework'} =  [
  'Core.Reset.css',
  'Core.Default.css',
  'Core.Form.css',
  'Core.Dialog.css',
  'Core.Tooltip.css',
  'Core.Login.css',
  'Core.Control.css',
  'Core.Table.css',
  'Core.TicketZoom.css',
  'Core.InputFields.css',
  'Core.Print.css',
  'thirdparty/fontawesome/font-awesome.css'
];
```

Loader::Customer::ResponsiveCSS###000-Framework
List of responsive CSS files to always be loaded for the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Loader::Customer::ResponsiveCSS'}->{'000-Framework'} =  [
  'Core.Responsive.css'
];
```

Loader::Customer::CommonJS###000-Framework
List of JS files to always be loaded for the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Loader::Customer::CommonJS'}->{'000-Framework'} =  [
  'thirdparty/jquery-2.1.4/jquery.js',
  'thirdparty/jquery-browser-detection/jquery-browser-detection.js',
  'thirdparty/jquery-validate-1.14.0/jquery.validate.js',
  'thirdparty/jquery-ui-1.11.4/jquery-ui.js',
  'thirdparty/stacktrace-0.6.4/stacktrace.js',
  'thirdparty/jquery-pubsub/pubsub.js',
  'thirdparty/jquery-jstree-3.1.1/jquery.jstree.js',
  'Core.Debug.js',
  'Core.Exception.js',
  'Core.Data.js',
  'Core.JSON.js',
  'Core.JavaScriptEnhancements.js',
  'Core.Config.js',
  'Core.App.js',
  'Core.App.Responsive.js',
  'Core.AJAX.js',
  'Core.UI.js',
  'Core.UI.InputFields.js',
  'Core.UI.Accessibility.js',
  'Core.UI.Dialog.js',
  'Core.UI.RichTextEditor.js',
  'Core.UI.Datepicker.js',
  'Core.UI.Popup.js',
  'Core.UI.TreeSelection.js',
  'Core.UI.Autocomplete.js',
  'Core.Form.js',
  'Core.Form.ErrorTooltips.js',
  'Core.Form.Validate.js',
  'Core.Customer.js',
  'Core.Customer.Responsive.js'
];
```

Loader::Customer::CommonJS###001-JQueryMigrate
List of JS files to always be loaded for the agent interface.

This setting is not active by default.

Default value:

```
$Self->{'Loader::Customer::CommonJS'}->{'001-JQueryMigrate'} =  [
  'thirdparty/jquery-migrate-1.2.1/jquery-migrate.js'
];
```

Framework → Core::WebUserAgent

WebUserAgent::Timeout
Sets the timeout (in seconds) for http/ftp downloads.

This setting can not be deactivated.

Default value:

```
$Self->{'WebUserAgent::Timeout'} =  '15';
```

WebUserAgent::Proxy
Defines the connections for http/ftp, via a proxy.

This setting is not active by default.

Default value:

```
$Self->{'WebUserAgent::Proxy'} =  'http://proxy.sn.no:8001/';
```

WebUserAgent::DisableSSLVerification
Turns off SSL certificate validation, for example if you use a transparent HTTPS proxy. Use at your own risk!

This setting can not be deactivated.

Default value:

```
$Self->{'WebUserAgent::DisableSSLVerification'} =  '0';
```

Framework → Crypt::PGP

PGP
Enables PGP support. When PGP support is enabled for signing and encrypting mail, it is HIGHLY recommended that the web server runs as the OTRS user. Otherwise, there will be problems with the privileges when accessing .gnupg folder.

This setting can not be deactivated.

Default value:

```
$Self->{'PGP'} =  '0';
```

PGP::Bin
Defines the path to PGP binary.

This setting can not be deactivated.

Default value:

```
$Self->{'PGP::Bin'} =  '/usr/bin/gpg';
```

PGP::Options
Sets the options for PGP binary.

This setting can not be deactivated.

Default value:

```
$Self->{'PGP::Options'} =  '--homedir /opt/otrs/.gnupg/ --batch --no-tty --yes';
```

PGP::Key::Password
Sets the password for private PGP key.

This setting can not be deactivated.

Default value:

```
$Self->{'PGP::Key::Password'} =  {
  '488A0B8F' => 'SomePassword',
  'D2DF79FA' => 'SomePassword'
};
```

PGP::TrustedNetwork

Set this to yes if you trust in all your public and private pgp keys, even if they are not certified with a trusted signature.

Default value:

```
$Self->{'PGP::TrustedNetwork'} =  '0';
```

PGP::Log

Configure your own log text for PGP.

This setting can not be deactivated.

Default value:

```
$Self->{'PGP::Log'} =  {
  'BADSIG' => 'The PGP signature with the keyid has not been verified successfully.',
  'ERRSIG' => 'It was not possible to check the PGP signature, this may be caused by a
 missing public key or an unsupported algorithm.',
  'EXPKEYSIG' => 'The PGP signature was made by an expired key.',
  'GOODSIG' => 'Good PGP signature.',
  'KEYREVOKED' => 'The PGP signature was made by a revoked key, this could mean that the
 signature is forged.',
  'NODATA' => 'No valid OpenPGP data found.',
  'NO_PUBKEY' => 'No public key found.',
  'REVKEYSIG' => 'The PGP signature was made by a revoked key, this could mean that the
 signature is forged.',
  'SIGEXPIRED' => 'The PGP signature is expired.',
  'SIG_ID' => 'Signature data.',
  'TRUST_UNDEFINED' => 'This key is not certified with a trusted signature!.',
  'VALIDSIG' => 'The PGP signature with the keyid is good.'
};
```

PGP::StoreDecryptedData

If this option is enabled, then the decrypted data will be stored in the database if they are displayed in AgentTicketZoom.

This setting can not be deactivated.

Default value:

```
$Self->{'PGP::StoreDecryptedData'} =  '1';
```

Framework → Crypt::SMIME

SMIME

Enables S/MIME support.

This setting can not be deactivated.

Default value:

```
$Self->{'SMIME'} =  '0';
```

SMIME::Bin

Defines the path to open ssl binary. It may need a HOME env ($ENV{HOME} = '/var/lib/wwwrun';).

This setting can not be deactivated.

Default value:

```
$Self->{'SMIME::Bin'} =  '/usr/bin/openssl';
```

SMIME::CertPath

Specifies the directory where SSL certificates are stored.

This setting can not be deactivated.

Default value:

```
$Self->{'SMIME::CertPath'} =  '/etc/ssl/certs';
```

SMIME::PrivatePath
Specifies the directory where private SSL certificates are stored.

This setting can not be deactivated.

Default value:

```
$Self->{'SMIME::PrivatePath'} =  '/etc/ssl/private';
```

SMIME::CacheTTL
Cache time in seconds for the SSL certificate attributes.

This setting can not be deactivated.

Default value:

```
$Self->{'SMIME::CacheTTL'} =  '86400';
```

SMIME::StoreDecryptedData
If this option is enabled, then the decrypted data will be stored in the database if they are displayed in AgentTicketZoom.

This setting can not be deactivated.

Default value:

```
$Self->{'SMIME::StoreDecryptedData'} =  '1';
```

SMIME::FetchFromCustomer
Enables fetch S/MIME from CustomerUser backend support.

This setting can not be deactivated.

Default value:

```
$Self->{'SMIME::FetchFromCustomer'} =  '0';
```

Framework → CustomerInformationCenter

AgentCustomerInformationCenter::MainMenu###010-EditCustomerID
Main menu registration.

This setting is not active by default.

Default value:

```
$Self->{'AgentCustomerInformationCenter::MainMenu'}->{'010-EditCustomerID'} =  {
 'Link' => '[% Env("Baselink")
%]Action=AdminCustomerCompany;Subaction=Change;CustomerID=[% Data.CustomerID | uri
%];Nav=0',
 'Name' => 'Edit customer company'
};
```

Framework → Frontend::Admin

Events###Package
List of all Package events to be displayed in the GUI.

This setting can not be deactivated.

Default value:

```
$Self->{'Events'}->{'Package'} =  [
  'PackageInstall',
  'PackageReinstall',
  'PackageUpgrade',
  'PackageUninstall'
];
```

Events###DynamicField
List of all DynamicField events to be displayed in the GUI.

This setting can not be deactivated.

Default value:

```
$Self->{'Events'}->{'DynamicField'} =  [
  'DynamicFieldAdd',
  'DynamicFieldUpdate',
  'DynamicFieldDelete'
];
```

Events###CustomerUser
List of all CustomerUser events to be displayed in the GUI.

This setting can not be deactivated.

Default value:

```
$Self->{'Events'}->{'CustomerUser'} =  [
  'CustomerUserAdd',
  'CustomerUserUpdate'
];
```

Events###CustomerCompany
List of all CustomerCompany events to be displayed in the GUI.

This setting can not be deactivated.

Default value:

```
$Self->{'Events'}->{'CustomerCompany'} =  [
  'CustomerCompanyAdd',
  'CustomerCompanyUpdate'
];
```

Framework → Frontend::Admin::AdminCustomerCompany

AdminCustomerCompany::RunInitialWildcardSearch
Runs an initial wildcard search of the existing customer company when accessing the AdminCustomerCompany module.

This setting can not be deactivated.

Default value:

```
$Self->{'AdminCustomerCompany::RunInitialWildcardSearch'} =  '1';
```

Framework → Frontend::Admin::AdminCustomerUser

AdminCustomerUser::RunInitialWildcardSearch
Runs an initial wildcard search of the existing customer users when accessing the AdminCustomerUser module.

This setting can not be deactivated.

Default value:

```
$Self->{'AdminCustomerUser::RunInitialWildcardSearch'} =  '1';
```

Framework → Frontend::Admin::AdminSelectBox

AdminSelectBox::AllowDatabaseModification

Controls if the admin is allowed to make changes to the database via AdminSelectBox.

This setting can not be deactivated.

Default value:

```
$Self->{'AdminSelectBox::AllowDatabaseModification'} =  '0';
```

Framework → Frontend::Admin::ModuleRegistration

Frontend::Module###Admin

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'Admin'} =  {
  'Description' => 'Admin Area.',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.css'
    ],
    'JavaScript' => [
      'Core.Agent.Admin.SysConfig.js'
    ]
  },
  'NavBar' => [
    {
      'AccessKey' => 'a',
      'Block' => 'ItemArea',
      'Description' => '',
      'Link' => 'Action=Admin',
      'LinkOption' => '',
      'Name' => 'Admin',
      'NavBar' => 'Admin',
      'Prio' => '10000',
      'Type' => 'Menu'
    }
  ],
  'NavBarModule' => {
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin'
  },
  'NavBarName' => 'Admin',
  'Title' => ''
};
```

Frontend::Module###AdminInit

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminInit'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'NavBarName' => '',
  'Title' => 'Init'
};
```

Frontend::Module###AdminUser

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminUser'} =  {
  'Description' => 'Create and manage agents.',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Agent',
    'Description' => 'Create and manage agents.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Agents',
    'Prio' => '100'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Agents'
};
```

Frontend::Module###AdminGroup

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminGroup'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Agent',
    'Description' => 'Create and manage groups.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Groups',
    'Prio' => '150'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Groups'
};
```

Frontend::Module###AdminUserGroup

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminUserGroup'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Agent',
    'Description' => 'Link agents to groups.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Agents <-> Groups',
    'Prio' => '200'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Agents <-> Groups'
};
```

Frontend::Module###AdminCustomerUser

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminCustomerUser'} =  {
  'Description' => 'Edit Customer Users.',
```

```
  'Group' => [
    'admin',
    'users'
  ],
  'GroupRo' => [
    ''
  ],
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.TicketAction.js'
    ]
  },
  'NavBar' => [
    {
      'AccessKey' => '',
      'Block' => 'ItemArea',
      'Description' => '',
      'Link' => 'Action=AdminCustomerUser;Nav=Agent',
      'LinkOption' => '',
      'Name' => 'Customer User Administration',
      'NavBar' => 'Customers',
      'Prio' => '9000',
      'Type' => ''
    }
  ],
  'NavBarModule' => {
    'Block' => 'Customer',
    'Description' => 'Create and manage customer users.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Customer User',
    'Prio' => '300'
  },
  'NavBarName' => 'Customers',
  'Title' => 'Customer Users'
};
```

Frontend::Module###AdminCustomerCompany

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminCustomerCompany'} =  {
  'Description' => 'Edit Customer Companies.',
  'Group' => [
    'admin',
    'users'
  ],
  'GroupRo' => [
    ''
  ],
  'NavBar' => [
    {
      'AccessKey' => '',
      'Block' => 'ItemArea',
      'Description' => '',
      'Link' => 'Action=AdminCustomerCompany;Nav=Agent',
      'LinkOption' => '',
      'Name' => 'Customer Administration',
      'NavBar' => 'Customers',
      'Prio' => '9100',
      'Type' => ''
    }
  ],
  'NavBarModule' => {
    'Block' => 'Customer',
    'Description' => 'Create and manage customers.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Customers',
    'Prio' => '310'
  },
  'NavBarName' => 'Customers',
  'Title' => 'Customer Companies'
```

```
};
```

Frontend::Module###AdminCustomerUserGroup

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminCustomerUserGroup'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Customer',
    'Description' => 'Link customer user to groups.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Customer User <-> Groups',
    'Prio' => '400'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Customers <-> Groups'
};
```

Frontend::Module###AdminCustomerUserService

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminCustomerUserService'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Customer',
    'Description' => 'Link customer user to services.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Customer User <-> Services',
    'Prio' => '500'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Customer User <-> Services'
};
```

Frontend::Module###AdminRole

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminRole'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Agent',
    'Description' => 'Create and manage roles.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Roles',
    'Prio' => '600'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Roles'
};
```

Frontend::Module###AdminRoleUser

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminRoleUser'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Agent',
    'Description' => 'Link agents to roles.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Agents <-> Roles',
    'Prio' => '700'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Agents <-> Roles'
};
```

Frontend::Module###AdminRoleGroup

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminRoleGroup'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Agent',
    'Description' => 'Link roles to groups.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Roles <-> Groups',
    'Prio' => '800'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Roles <-> Groups'
};
```

Frontend::Module###AdminSMIME

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminSMIME'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Email',
    'Description' => 'Manage S/MIME certificates for email encryption.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'S/MIME Certificates',
    'Prio' => '1100'
  },
  'NavBarName' => 'Admin',
  'Title' => 'S/MIME Management'
};
```

Frontend::Module###AdminPGP

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminPGP'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Email',
```

```
    'Description' => 'Manage PGP keys for email encryption.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'PGP Keys',
    'Prio' => '1200'
  },
  'NavBarName' => 'Admin',
  'Title' => 'PGP Key Management'
};
```

Frontend::Module###AdminMailAccount

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminMailAccount'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Email',
    'Description' => 'Manage POP3 or IMAP accounts to fetch email from.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'PostMaster Mail Accounts',
    'Prio' => '100'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Mail Accounts'
};
```

Frontend::Module###AdminPostMasterFilter

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminPostMasterFilter'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Email',
    'Description' => 'Filter incoming emails.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'PostMaster Filters',
    'Prio' => '200'
  },
  'NavBarName' => 'Admin',
  'Title' => 'PostMaster Filters'
};
```

Frontend::Module###AdminEmail

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminEmail'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'System',
    'Description' => 'Send notifications to users.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Admin Notification',
    'Prio' => '400'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Admin Notification'
```

```
};
```

Frontend::Module###AdminSession

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminSession'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'System',
    'Description' => 'Manage existing sessions.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Session Management',
    'Prio' => '500'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Session Management'
};
```

Frontend::Module###AdminPerformanceLog

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminPerformanceLog'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.PerformanceLog.css'
    ]
  },
  'NavBarModule' => {
    'Block' => 'System',
    'Description' => 'View performance benchmark results.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Performance Log',
    'Prio' => '550'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Performance Log'
};
```

Frontend::Module###AdminRegistration

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminRegistration'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.Registration.css'
    ]
  },
  'NavBarModule' => {
    'Block' => 'System',
    'Description' => 'Manage system registration.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'System Registration',
    'Prio' => '350'
```

```
  },
  'NavBarName' => 'Admin',
  'Title' => 'System Registration'
};
```

Frontend::Module###AdminOTRSBusiness

Frontend module registration for the agent interface.

Default value:

```
 $Self->{'Frontend::Module'}->{'AdminOTRSBusiness'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.OTRSBusiness.css'
    ]
  },
  'NavBarModule' => {
    'Block' => 'System',
    'Description' => 'Deploy and manage OTRS Business Solution™.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'OTRS Business Solution™',
    'Prio' => '360'
  },
  'NavBarName' => 'Admin',
  'Title' => 'OTRS Business Solution™'
};
```

Frontend::Module###AdminSupportDataCollector

Frontend module registration for the agent interface.

Default value:

```
 $Self->{'Frontend::Module'}->{'AdminSupportDataCollector'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.SupportDataCollector.css'
    ]
  },
  'NavBarModule' => {
    'Block' => 'System',
    'Description' => 'Manage support data.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Support Data Collector',
    'Prio' => '370'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Support Data Collector'
};
```

Frontend::Module###AdminCloudServices

Frontend module registration for the agent interface.

Default value:

```
 $Self->{'Frontend::Module'}->{'AdminCloudServices'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.CloudServices.css'
```

```
    ]
  },
  'NavBarModule' => {
    'Block' => 'System',
    'Description' => 'Manage OTRS Group cloud services.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Cloud Services',
    'Prio' => '380'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Cloud Services'
};
```

Frontend::Module###AdminLog

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminLog'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'System',
    'Description' => 'View system log messages.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'System Log',
    'Prio' => '600'
  },
  'NavBarName' => 'Admin',
  'Title' => 'System Log'
};
```

Frontend::Module###AdminSelectBox

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminSelectBox'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'System',
    'Description' => 'Execute SQL statements.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'SQL Box',
    'Prio' => '700'
  },
  'NavBarName' => 'Admin',
  'Title' => 'SQL Box'
};
```

Frontend::Module###AdminPackageManager

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminPackageManager'} =  {
  'Description' => 'Software Package Manager.',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'System',
    'Description' => 'Update and extend your system with software packages.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Package Manager',
```

```
    'Prio' => '1000'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Package Manager'
};
```

Frontend::Module###AdminSystemMaintenance

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminSystemMaintenance'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'System',
    'Description' => 'Schedule a maintenance period.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'System Maintenance',
    'Prio' => '501'
  },
  'NavBarName' => 'Admin',
  'Title' => 'System Maintenance'
};
```

Frontend::Module###AdminCloudServiceSupportDataCollector

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminCloudServiceSupportDataCollector'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.CloudService.SupportDataCollector.css'
    ]
  },
  'Title' => 'Support data collector'
};
```

Framework → Frontend::Agent

AgentLogo

The logo shown in the header of the agent interface. The URL to the image can be a relative URL to the skin image directory, or a full URL to a remote web server.

Default value:

```
$Self->{'AgentLogo'} =  {
  'StyleHeight' => '85px',
  'StyleRight' => '38px',
  'StyleTop' => '4px',
  'StyleWidth' => '270px',
  'URL' => 'skins/Agent/default/img/logo_bg.png'
};
```

AgentLogoCustom###default

The logo shown in the header of the agent interface for the skin "default". See "AgentLogo" for further description.

This setting is not active by default.

Default value:

```
$Self->{'AgentLogoCustom'}->{'default'} =  {
  'StyleHeight' => '67px',
  'StyleRight' => '38px',
  'StyleTop' => '4px',
  'StyleWidth' => '270px',
  'URL' => 'skins/Agent/default/img/logo_bg.png'
};
```

AgentLogoCustom###slim

The logo shown in the header of the agent interface for the skin "slim". See "AgentLogo" for further description.

This setting is not active by default.

Default value:

```
$Self->{'AgentLogoCustom'}->{'slim'} =  {
  'StyleHeight' => '67px',
  'StyleRight' => '38px',
  'StyleTop' => '4px',
  'StyleWidth' => '270px',
  'URL' => 'skins/Agent/default/img/logo_bg.png'
};
```

AgentLogoCustom###ivory

The logo shown in the header of the agent interface for the skin "ivory". See "AgentLogo" for further description.

This setting is not active by default.

Default value:

```
$Self->{'AgentLogoCustom'}->{'ivory'} =  {
  'StyleHeight' => '67px',
  'StyleRight' => '38px',
  'StyleTop' => '4px',
  'StyleWidth' => '270px',
  'URL' => 'skins/Agent/default/img/logo_bg.png'
};
```

AgentLogoCustom###ivory-slim

The logo shown in the header of the agent interface for the skin "ivory-slim". See "AgentLogo" for further description.

This setting is not active by default.

Default value:

```
$Self->{'AgentLogoCustom'}->{'ivory-slim'} =  {
  'StyleHeight' => '67px',
  'StyleRight' => '38px',
  'StyleTop' => '4px',
  'StyleWidth' => '270px',
  'URL' => 'skins/Agent/default/img/logo_bg.png'
};
```

AgentLoginLogo

The logo shown on top of the login box of the agent interface. The URL to the image must be relative URL to the skin image directory.

This setting is not active by default.

Default value:

```
$Self->{'AgentLoginLogo'} =  {
  'StyleHeight' => '100px',
  'URL' => 'skins/Agent/default/img/loginlogo_default.png'
```

```
};
```

LoginURL
Defines an alternate URL, where the login link refers to.

This setting is not active by default.

Default value:

```
$Self->{'LoginURL'} =  'http://host.example.com/login.html';
```

LogoutURL
Defines an alternate URL, where the logout link refers to.

This setting is not active by default.

Default value:

```
$Self->{'LogoutURL'} =  'http://host.example.com/thanks-for-using-otrs.html';
```

PreApplicationModule###AgentInfo
Defines a useful module to load specific user options or to display news.

This setting is not active by default.

Default value:

```
$Self->{'PreApplicationModule'}->{'AgentInfo'} =  'Kernel::Modules::AgentInfo';
```

InfoKey
Defines the key to be checked with Kernel::Modules::AgentInfo module. If this user preferences key is true, the message is accepted by the system.

This setting can not be deactivated.

Default value:

```
$Self->{'InfoKey'} =  'wpt22';
```

InfoFile
File that is displayed in the Kernel::Modules::AgentInfo module, if located under Kernel/Output/HTML/Templates/Standard/AgentInfo.tt.

This setting can not be deactivated.

Default value:

```
$Self->{'InfoFile'} =  'AgentInfo';
```

LostPassword
Activates lost password feature for agents, in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'LostPassword'} =  '1';
```

ShowMotd
Shows the message of the day on login screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'ShowMotd'} =  '0';
```

NotificationSubjectLostPasswordToken

Defines the subject for notification mails sent to agents, with token about new requested password.

This setting can not be deactivated.

Default value:

```
$Self->{'NotificationSubjectLostPasswordToken'} =  'New OTRS password request';
```

NotificationBodyLostPasswordToken

Defines the body text for notification mails sent to agents, with token about new requested password (after using this link the new password will be sent).

This setting can not be deactivated.

Default value:

```
$Self->{'NotificationBodyLostPasswordToken'} =  'Hi <OTRS_USERFIRSTNAME>,

You or someone impersonating you has requested to change your OTRS
password.

If you want to do this, click on the link below. You will receive another email
 containing the password.

<OTRS_CONFIG_HttpType>://<OTRS_CONFIG_FQDN>/<OTRS_CONFIG_ScriptAlias>index.pl?
Action=LostPassword;Token=<OTRS_TOKEN>

If you did not request a new password, please ignore this email.
            ';
```

NotificationSubjectLostPassword

Defines the subject for notification mails sent to agents, about new password.

This setting can not be deactivated.

Default value:

```
$Self->{'NotificationSubjectLostPassword'} =  'New OTRS password';
```

NotificationBodyLostPassword

Defines the body text for notification mails sent to agents, about new password (after using this link the new password will be sent).

This setting can not be deactivated.

Default value:

```
$Self->{'NotificationBodyLostPassword'} =  'Hi <OTRS_USERFIRSTNAME>,


Here\'s your new OTRS password.

New password: <OTRS_NEWPW>

You can log in via the following URL:

<OTRS_CONFIG_HttpType>://<OTRS_CONFIG_FQDN>/<OTRS_CONFIG_ScriptAlias>index.pl
            ';
```

OpenMainMenuOnHover

If enabled, the first level of the main menu opens on mouse hover (instead of click only).

This setting can not be deactivated.

Default value:

```
$Self->{'OpenMainMenuOnHover'} =  '0';
```

FirstnameLastnameOrder
Specifies the order in which the firstname and the lastname of agents will be displayed.

This setting can not be deactivated.

Default value:

```
$Self->{'FirstnameLastnameOrder'} =  '0';
```

Loader::Agent::Skin###000-default
Default skin for the agent interface.

Default value:

```
$Self->{'Loader::Agent::Skin'}->{'000-default'} =  {
  'Description' => 'This is the default orange - black skin.',
  'HomePage' => 'www.otrs.org',
  'InternalName' => 'default',
  'VisibleName' => 'Default'
};
```

Loader::Agent::Skin###001-slim
Default skin for the agent interface (slim version).

Default value:

```
$Self->{'Loader::Agent::Skin'}->{'001-slim'} =  {
  'Description' => '"Slim" skin which tries to save screen space for power users.',
  'HomePage' => 'www.otrs.org',
  'InternalName' => 'slim',
  'VisibleName' => 'Default (Slim)'
};
```

Loader::Agent::Skin###001-ivory
Balanced white skin by Felix Niklas.

Default value:

```
$Self->{'Loader::Agent::Skin'}->{'001-ivory'} =  {
  'Description' => 'Balanced white skin by Felix Niklas.',
  'HomePage' => 'www.felixniklas.de',
  'InternalName' => 'ivory',
  'VisibleName' => 'Ivory'
};
```

Loader::Agent::Skin###001-ivory-slim
Balanced white skin by Felix Niklas (slim version).

Default value:

```
$Self->{'Loader::Agent::Skin'}->{'001-ivory-slim'} =  {
  'Description' => 'Balanced white skin by Felix Niklas (slim version).',
  'HomePage' => 'www.felixniklas.de',
  'InternalName' => 'ivory-slim',
  'VisibleName' => 'Ivory (Slim)'
};
```

Loader::Agent::DefaultSelectedSkin
The agent skin's InternalName which should be used in the agent interface. Please check the available skins in Frontend::Agent::Skins.

This setting can not be deactivated.

Default value:

```
$Self->{'Loader::Agent::DefaultSelectedSkin'} =  'default';
```

Loader::Agent::DefaultSelectedSkin::HostBased

It is possible to configure different skins, for example to distinguish between diferent agents, to be used on a per-domain basis within the application. Using a regular expression (regex), you can configure a Key/Content pair to match a domain. The value in "Key" should match the domain, and the value in "Content" should be a valid skin on your system. Please see the example entries for the proper form of the regex.

This setting is not active by default.

Default value:

```
$Self->{'Loader::Agent::DefaultSelectedSkin::HostBased'} =  {
  'host1\\.example\\.com' => 'SomeSkin1',
  'host2\\.example\\.com' => 'SomeSkin2'
};
```

AutoComplete::Agent###Default

Defines the config options for the autocompletion feature.

Default value:

```
$Self->{'AutoComplete::Agent'}->{'Default'} =  {
  'AutoCompleteActive' => '1',
  'ButtonText' => 'Search',
  'MaxResultsDisplayed' => '20',
  'MinQueryLength' => '2',
  'QueryDelay' => '100'
};
```

AutoComplete::Agent###CustomerSearch

Defines the config options for the autocompletion feature.

Default value:

```
$Self->{'AutoComplete::Agent'}->{'CustomerSearch'} =  {
  'AutoCompleteActive' => '1',
  'ButtonText' => 'Search Customer',
  'MaxResultsDisplayed' => '20',
  'MinQueryLength' => '2',
  'QueryDelay' => '100'
};
```

AutoComplete::Agent###UserSearch

Defines the config options for the autocompletion feature.

Default value:

```
$Self->{'AutoComplete::Agent'}->{'UserSearch'} =  {
  'AutoCompleteActive' => '1',
  'ButtonText' => 'Search User',
  'MaxResultsDisplayed' => '20',
  'MinQueryLength' => '2',
  'QueryDelay' => '100'
};
```

PossibleNextActions

Defines the list of possible next actions on an error screen, a full path is required, then is possible to add external links if needed.

Default value:

```
$Self->{'PossibleNextActions'} =  {
  'Go to dashboard!' => '[% Env(\'CGIHandle\') %]?Action=AgentDashboard'
};
```

ModernizeFormFields

Use new type of select and autocomplete fields in agent interface, where applicable (InputFields).

This setting can not be deactivated.

Default value:

```
$Self->{'ModernizeFormFields'} =  '1';
```

Framework → Frontend::Agent::Auth::TwoFactor

AuthTwoFactorModule

Defines the two-factor module to authenticate agents.

This setting is not active by default.

Default value:

```
$Self->{'AuthTwoFactorModule'} =
'Kernel::System::Auth::TwoFactor::GoogleAuthenticator';
```

AuthTwoFactorModule::SecretPreferencesKey

Defines the agent preferences key where the shared secret key is stored.

This setting can not be deactivated.

Default value:

```
$Self->{'AuthTwoFactorModule::SecretPreferencesKey'} =
'UserGoogleAuthenticatorSecretKey';
```

AuthTwoFactorModule::AllowEmptySecret

Defines if agents should be allowed to login if they have no shared secret stored in their preferences and therefore are not using two-factor authentication.

Default value:

```
$Self->{'AuthTwoFactorModule::AllowEmptySecret'} =  '1';
```

AuthTwoFactorModule::AllowPreviousToken

Defines if the previously valid token should be accepted for authentication. This is slightly less secure but gives users 30 seconds more time to enter their one-time password.

Default value:

```
$Self->{'AuthTwoFactorModule::AllowPreviousToken'} =  '1';
```

Framework → Frontend::Agent::Dashboard

AgentCustomerInformationCenter::Backend###0600-CIC-CustomerCompanyInformation

Parameters for the dashboard backend of the customer company information of the agent interface . "Group" is used to restrict the access to the plugin (e. g. Group: admin;group1;group2;). "Default" determines if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTLLocal" is the cache time in minutes for the plugin.

Default value:

```
$Self->{'AgentCustomerInformationCenter::Backend'}->{'0600-CIC-
CustomerCompanyInformation'} =  {
```

```
  'Attributes' => '',
  'Block' => 'ContentSmall',
  'Default' => '1',
  'Description' => 'Customer Information',
  'Group' => '',
  'Module' => 'Kernel::Output::HTML::Dashboard::CustomerCompanyInformation',
  'Title' => 'Customer Information'
};
```

DashboardBackend###0000-ProductNotify

Defines the parameters for the dashboard backend. "Group" is used to restrict access to the plugin (e. g. Group: admin;group1;group2;). "Default" indicates if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTLLocal" defines the cache expiration period in minutes for the plugin.

Default value:

```
$Self->{'DashboardBackend'}->{'0000-ProductNotify'} =  {
  'Block' => 'ContentLarge',
  'CacheTTLLocal' => '1440',
  'Default' => '1',
  'Description' => 'News about OTRS releases!',
  'Group' => 'admin',
  'Module' => 'Kernel::Output::HTML::Dashboard::ProductNotify',
  'Title' => 'Product News'
};
```

DashboardBackend###0390-UserOutOfOffice

Defines the parameters for the dashboard backend. "Limit" defines the number of entries displayed by default. "Group" is used to restrict access to the plugin (e. g. Group: admin;group1;group2;). "Default" indicates if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTLLocal" defines the cache expiration period in minutes for the plugin.

Default value:

```
$Self->{'DashboardBackend'}->{'0390-UserOutOfOffice'} =  {
  'Block' => 'ContentSmall',
  'CacheTTLLocal' => '5',
  'Default' => '1',
  'Description' => '',
  'Group' => '',
  'IdleMinutes' => '60',
  'Limit' => '10',
  'Module' => 'Kernel::Output::HTML::Dashboard::UserOutOfOffice',
  'SortBy' => 'UserFullname',
  'Title' => 'Out Of Office'
};
```

DashboardBackend###0400-UserOnline

Defines the parameters for the dashboard backend. "Limit" defines the number of entries displayed by default. "Group" is used to restrict access to the plugin (e. g. Group: admin;group1;group2;). "Default" indicates if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTLLocal" defines the cache expiration period in minutes for the plugin.

Default value:

```
$Self->{'DashboardBackend'}->{'0400-UserOnline'} =  {
  'Block' => 'ContentSmall',
  'CacheTTLLocal' => '5',
  'Default' => '0',
  'Description' => '',
  'Filter' => 'Agent',
  'Group' => '',
  'IdleMinutes' => '60',
  'Limit' => '10',
  'Module' => 'Kernel::Output::HTML::Dashboard::UserOnline',
```

```
  'ShowEmail' => '0',
  'SortBy' => 'UserFullname',
  'Title' => 'Online'
};
```

DashboardBackend###0405-News

Defines the parameters for the dashboard backend. "Limit" defines the number of entries displayed by default. "Group" is used to restrict access to the plugin (e. g. Group: admin;group1;group2;). "Default" indicates if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTL" indicates the cache expiration period in minutes for the plugin.

Default value:

```
$Self->{'DashboardBackend'}->{'0405-News'} =  {
  'Block' => 'ContentSmall',
  'CacheTTL' => '360',
  'Default' => '1',
  'Description' => '',
  'Group' => '',
  'Limit' => '6',
  'Module' => 'Kernel::Output::HTML::Dashboard::News',
  'Title' => 'OTRS News'
};
```

DashboardBackend###0410-RSS

Defines the parameters for the dashboard backend. "Limit" defines the number of entries displayed by default. "Group" is used to restrict access to the plugin (e. g. Group: admin;group1;group2;). "Default" indicates if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTL" indicates the cache expiration period in minutes for the plugin.

This setting is not active by default.

Default value:

```
$Self->{'DashboardBackend'}->{'0410-RSS'} =  {
  'Block' => 'ContentSmall',
  'CacheTTL' => '360',
  'Default' => '1',
  'Description' => '',
  'Group' => '',
  'Limit' => '6',
  'Module' => 'Kernel::Output::HTML::Dashboard::RSS',
  'Title' => 'Custom RSS Feed',
  'URL' => 'http://www.otrs.com/en/rss.xml',
  'URL_de' => 'http://www.otrs.com/de/rss.xml',
  'URL_es' => 'http://www.otrs.com/es/rss.xml',
  'URL_nl' => 'http://www.otrs.com/nl/rss.xml',
  'URL_ru' => 'http://www.otrs.com/ru/rss.xml',
  'URL_zh' => 'http://www.otrs.com/cn/rss.xml'
};
```

DashboardBackend###0420-CmdOutput

Defines the parameters for the dashboard backend. "Cmd" is used to specify command with parameters. "Group" is used to restrict access to the plugin (e. g. Group: admin;group1;group2;). "Default" indicates if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTL" indicates the cache expiration period in minutes for the plugin.

This setting is not active by default.

Default value:

```
$Self->{'DashboardBackend'}->{'0420-CmdOutput'} =  {
  'Block' => 'ContentSmall',
  'CacheTTL' => '60',
```

```
  'Cmd' => '/bin/echo Configure me please.',
  'Default' => '0',
  'Description' => '',
  'Group' => '',
  'Module' => 'Kernel::Output::HTML::Dashboard::CmdOutput',
  'Title' => 'Sample command output'
};
```

DashboardBackend###0200-Image

Defines the parameters for the dashboard backend. "Group" is used to restrict access to the plugin (e. g. Group: admin;group1;group2;). "Default" indicates if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTL" indicates the cache expiration period in minutes for the plugin.

This setting is not active by default.

Default value:

```
$Self->{'DashboardBackend'}->{'0200-Image'} =  {
  'Block' => 'ContentLarge',
  'Default' => '1',
  'Description' => 'Some picture description!',
  'Group' => '',
  'Height' => '140',
  'Link' => 'http://otrs.org/',
  'LinkTitle' => 'http://otrs.org/',
  'Module' => 'Kernel::Output::HTML::Dashboard::Image',
  'Title' => 'A picture',
  'URL' => 'http://www.otrs.com/wp-uploads//2013/10/OTRS_Logo-300x170.png',
  'Width' => '198'
};
```

DashboardBackend###0210-MOTD

Shows the message of the day (MOTD) in the agent dashboard. "Group" is used to restrict access to the plugin (e. g. Group: admin;group1;group2;). "Default" indicates if the plugin is enabled by default or if the user needs to enable it manually.

This setting is not active by default.

Default value:

```
$Self->{'DashboardBackend'}->{'0210-MOTD'} =  {
  'Block' => 'ContentLarge',
  'Default' => '1',
  'Group' => '',
  'Module' => 'Kernel::Output::HTML::Dashboard::MOTD',
  'Title' => 'Message of the Day'
};
```

DashboardBackend###0300-IFrame

Defines the parameters for the dashboard backend. "Group" is used to restrict access to the plugin (e. g. Group: admin;group1;group2;). "Default" indicates if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTL" indicates the cache expiration period in minutes for the plugin.

This setting is not active by default.

Default value:

```
$Self->{'DashboardBackend'}->{'0300-IFrame'} =  {
  'Align' => 'left',
  'Block' => 'ContentLarge',
  'Default' => '1',
  'Description' => 'Some description!',
  'Frameborder' => '1',
  'Group' => '',
  'Height' => '800',
  'Link' => 'http://otrs.org/',
```

```
  'LinkTitle' => 'OTRS.org/',
  'Marginheight' => '5',
  'Marginwidth' => '5',
  'Module' => 'Kernel::Output::HTML::Dashboard::IFrame',
  'Scrolling' => 'auto',
  'Title' => 'A Website',
  'URL' => 'http://www.otrs.org/',
  'Width' => '1024'
};
```

AgentCustomerInformationCenter::Backend###0050-CIC-CustomerUserList

Parameters for the dashboard backend of the customer user list overview of the agent interface . "Limit" is the number of entries shown by default. "Group" is used to restrict the access to the plugin (e. g. Group: admin;group1;group2;). "Default" determines if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTLLocal" is the cache time in minutes for the plugin.

Default value:

```
$Self->{'AgentCustomerInformationCenter::Backend'}->{'0050-CIC-CustomerUserList'} =  {
  'Attributes' => '',
  'Block' => 'ContentLarge',
  'CacheTTLLocal' => '0.5',
  'Default' => '1',
  'Description' => 'All customer users of a CustomerID',
  'Group' => '',
  'Limit' => '10',
  'Module' => 'Kernel::Output::HTML::Dashboard::CustomerUserList',
  'Permission' => 'ro',
  'Title' => 'Customer Users'
};
```

Framework → Frontend::Agent::LinkObject

Frontend::AgentLinkObject::WildcardSearch

Starts a wildcard search of the active object after the link object mask is started.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::AgentLinkObject::WildcardSearch'} =  '0';
```

Framework → Frontend::Agent::ModuleMetaHead

Frontend::HeaderMetaModule###100-Refresh

Defines the module to generate code for periodic page reloads.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::HeaderMetaModule'}->{'100-Refresh'} =  {
  'Module' => 'Kernel::Output::HTML::HeaderMeta::Refresh'
};
```

Framework → Frontend::Agent::ModuleNotify

Frontend::NotifyModule###100-OTRSBusiness

Defines the module to display a notification in different interfaces on different occasions for OTRS Business Solution™.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::NotifyModule'}->{'100-OTRSBusiness'} =  {
  'Group' => 'admin',
  'Module' => 'Kernel::Output::HTML::Notification::AgentOTRSBusiness'
};
```

Frontend::NotifyModule###200-UID-Check

Defines the module to display a notification in the agent interface, if the system is used by the admin user (normally you shouldn't work as admin).

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::NotifyModule'}->{'200-UID-Check'} =  {
  'Module' => 'Kernel::Output::HTML::Notification::UIDCheck'
};
```

Frontend::NotifyModule###250-AgentSessionLimit

Defines the module to display a notification in the agent interface, if the agent session limit prior warning is reached.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::NotifyModule'}->{'250-AgentSessionLimit'} =  {
  'Module' => 'Kernel::Output::HTML::Notification::AgentSessionLimit'
};
```

Frontend::NotifyModule###300-ShowAgentOnline

Defines the module that shows all the currently logged in agents in the agent interface.

This setting is not active by default.

Default value:

```
$Self->{'Frontend::NotifyModule'}->{'300-ShowAgentOnline'} =  {
  'IdleMinutes' => '60',
  'Module' => 'Kernel::Output::HTML::Notification::AgentOnline',
  'ShowEmail' => '1'
};
```

Frontend::NotifyModule###400-ShowCustomerOnline

Defines the module that shows all the currently logged in customers in the agent interface.

This setting is not active by default.

Default value:

```
$Self->{'Frontend::NotifyModule'}->{'400-ShowCustomerOnline'} =  {
  'IdleMinutes' => '60',
  'Module' => 'Kernel::Output::HTML::Notification::CustomerOnline',
  'ShowEmail' => '1'
};
```

Frontend::NotifyModule###500-OutofOffice-Check

Defines the module to display a notification in the agent interface, if the agent is logged in while having out-of-office active.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::NotifyModule'}->{'500-OutofOffice-Check'} =  {
  'Module' => 'Kernel::Output::HTML::Notification::OutofOfficeCheck'
};
```

Frontend::NotifyModule###600-SystemMaintenance-Check

Defines the module to display a notification in the agent interface, if the agent is logged in while having system maintenance active.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::NotifyModule'}->{'600-SystemMaintenance-Check'} =  {
  'Module' => 'Kernel::Output::HTML::Notification::SystemMaintenanceCheck'
};
```

Frontend::NotifyModule###900-Generic

Defines the module that shows a generic notification in the agent interface. Either "Text" - if configured - or the contents of "File" will be displayed.

This setting is not active by default.

Default value:

```
$Self->{'Frontend::NotifyModule'}->{'900-Generic'} =  {
  'File' => '<OTRS_CONFIG_Home>/var/notify.txt',
  'Link' => 'http://www.otrs.com',
  'Module' => 'Kernel::Output::HTML::Notification::Generic',
  'Priority' => 'Warning',
  'Text' => 'The OTRS Website'
};
```

Framework → Frontend::Agent::ModuleRegistration

Frontend::Module###Logout

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'Logout'} =  {
  'Description' => 'Logout',
  'NavBarName' => '',
  'Title' => ''
};
```

Frontend::Module###AgentDashboard

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentDashboard'} =  {
  'Description' => 'Agent Dashboard',
  'Loader' => {
    'CSS' => [
      'Core.Agent.Dashboard.css',
      'Core.AllocationList.css',
      'thirdparty/fullcalendar-2.4.0/fullcalendar.min.css',
      'thirdparty/nvd3-1.7.1/nv.d3.css'
    ],
    'JavaScript' => [
      'thirdparty/momentjs-2.10.6/moment.min.js',
      'thirdparty/fullcalendar-2.4.0/fullcalendar.min.js',
      'thirdparty/d3-3.5.6/d3.min.js',
      'thirdparty/nvd3-1.7.1/nvd3.min.js',
      'thirdparty/nvd3-1.7.1/models/OTRSLineChart.js',
      'thirdparty/nvd3-1.7.1/models/OTRSMultiBarChart.js',
      'thirdparty/nvd3-1.7.1/models/OTRSStackedAreaChart.js',
      'thirdparty/canvg-1.4/rgbcolor.js',
      'thirdparty/canvg-1.4/StackBlur.js',
      'thirdparty/canvg-1.4/canvg.js',
      'thirdparty/StringView-8/stringview.js',
      'Core.UI.AdvancedChart.js',
      'Core.UI.AllocationList.js',
```

```
      'Core.Agent.TableFilters.js',
      'Core.Agent.Dashboard.js'
    ]
  },
  'NavBar' => [
    {
      'AccessKey' => 'd',
      'Block' => 'ItemArea',
      'Description' => '',
      'Link' => 'Action=AgentDashboard',
      'LinkOption' => '',
      'Name' => 'Dashboard',
      'NavBar' => 'Dashboard',
      'Prio' => '50',
      'Type' => 'Menu'
    }
  ],
  'NavBarName' => 'Dashboard',
  'Title' => ''
};
```

Frontend::Module###AgentCustomerInformationCenter

Frontend module registration for the agent interface.

Default value:

```
 $Self->{'Frontend::Module'}->{'AgentCustomerInformationCenter'} =  {
  'Description' => 'Customer Information Center.',
  'Loader' => {
    'CSS' => [
      'Core.AllocationList.css'
    ],
    'JavaScript' => [
      'Core.UI.AllocationList.js',
      'Core.Agent.Dashboard.js',
      'Core.Agent.TableFilters.js'
    ]
  },
  'NavBar' => [
    {
      'AccessKey' => 'c',
      'Block' => 'ItemArea',
      'Description' => '',
      'Link' => 'Action=AgentCustomerInformationCenter',
      'LinkOption' => 'onclick="window.setTimeout(function()
{Core.Agent.CustomerInformationCenterSearch.OpenSearchDialog();}, 0); return false;"',
      'Name' => 'Customer Information Center',
      'NavBar' => 'Customers',
      'Prio' => '50',
      'Type' => ''
    },
    {
      'AccessKey' => '',
      'Block' => 'ItemArea',
      'Description' => '',
      'Link' => 'Action=AgentCustomerInformationCenter',
      'LinkOption' => '',
      'Name' => 'Customers',
      'NavBar' => 'Customers',
      'Prio' => '60',
      'Type' => 'Menu'
    }
  ],
  'NavBarName' => 'Customers',
  'Title' => ''
};
```

Frontend::Module###AgentCustomerInformationCenterSearch

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentCustomerInformationCenterSearch'} =  {
  'Description' => 'Customer Information Center Search.',
  'Title' => ''
};
```

Frontend::Module###AgentPreferences
Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentPreferences'} =  {
  'Description' => 'Agent Preferences.',
  'Loader' => {
    'CSS' => [
      'Core.Agent.Preferences.css'
    ]
  },
  'NavBarName' => 'Preferences',
  'Title' => ''
};
```

Frontend::Module###PictureUpload
Frontend module registration for the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::Module'}->{'PictureUpload'} =  {
  'Description' => 'Picture upload module.',
  'NavBarName' => 'Ticket',
  'Title' => 'Picture Upload'
};
```

Frontend::Module###AgentSpelling
Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentSpelling'} =  {
  'Description' => 'Spell checker.',
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.TicketAction.js'
    ]
  },
  'NavBarName' => '',
  'Title' => 'Spell Checker'
};
```

Frontend::Module###SpellingInline
Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'SpellingInline'} =  {
  'Description' => 'Spell checker.',
  'NavBarName' => '',
  'Title' => 'Spell Checker'
};
```

Frontend::Module###AgentBook
Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentBook'} =  {
  'Description' => 'Address book of CustomerUser sources.',
```

```
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.CustomerSearch.js',
      'Core.Agent.TicketAction.js'
    ]
  },
  'NavBarName' => '',
  'Title' => 'Address Book'
};
```

Frontend::Module###AgentLinkObject

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentLinkObject'} =  {
  'Description' => 'Link Object.',
  'NavBarName' => '',
  'Title' => 'Link Object'
};
```

Frontend::Module###AgentInfo

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentInfo'} =  {
  'Description' => 'Generic Info module.',
  'NavBarName' => '',
  'Title' => 'Info'
};
```

Frontend::Module###AgentSearch

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentSearch'} =  {
  'Description' => 'Global Search Module.',
  'NavBarName' => '',
  'Title' => 'Search'
};
```

Frontend::Module###AgentOTRSBusiness

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentOTRSBusiness'} =  {
  'Description' => 'Agent',
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.OTRSBusiness.css'
    ]
  },
  'NavBarName' => '',
  'Title' => 'OTRS Business Solution™'
};
```

CustomerFrontend::Module###SpellingInline

Frontend module registration for the customer interface.

Default value:

```
$Self->{'CustomerFrontend::Module'}->{'SpellingInline'} =  {
  'Description' => 'Spell checker.',
  'NavBarName' => '',
  'Title' => 'Spell Checker'
```

```
};
```

Frontend::Module###AgentHTMLReference

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentHTMLReference'} =  {
  'Description' => 'HTML Reference.',
  'Group' => [
    'users'
  ],
  'GroupRo' => [
    'users'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.HTMLReference.css'
    ]
  },
  'NavBarName' => '',
  'Title' => 'HTML Reference'
};
```

Frontend::Module###AgentStatistics

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentStatistics'} =  {
  'Description' => '',
  'Group' => [
    'stats'
  ],
  'GroupRo' => [
    'stats'
  ],
  'Loader' => {
    'CSS' => [
      'thirdparty/nvd3-1.7.1/nv.d3.css',
      'Core.Agent.Statistics.css'
    ],
    'JavaScript' => [
      'thirdparty/d3-3.5.6/d3.min.js',
      'thirdparty/nvd3-1.7.1/nvd3.min.js',
      'thirdparty/nvd3-1.7.1/models/OTRSLineChart.js',
      'thirdparty/nvd3-1.7.1/models/OTRSMultiBarChart.js',
      'thirdparty/nvd3-1.7.1/models/OTRSStackedAreaChart.js',
      'thirdparty/canvg-1.4/rgbcolor.js',
      'thirdparty/canvg-1.4/StackBlur.js',
      'thirdparty/canvg-1.4/canvg.js',
      'thirdparty/StringView-8/stringview.js',
      'Core.Agent.Statistics.js',
      'Core.UI.AdvancedChart.js'
    ]
  },
  'NavBar' => [
    {
      'AccessKey' => '',
      'Block' => 'ItemArea',
      'Description' => '',
      'Link' => 'Action=AgentStatistics;Subaction=Overview',
      'LinkOption' => '',
      'Name' => 'Reports',
      'NavBar' => 'Reports',
      'Prio' => '8500',
      'Type' => 'Menu'
    },
    {
      'AccessKey' => '',
      'Block' => '',
```

```
      'Description' => '',
      'GroupRo' => [
        'stats'
      ],
      'Link' => 'Action=AgentStatisticsReports;Subaction=Overview',
      'LinkOption' => 'class="OTRSBusinessRequired"',
      'Name' => 'Reports (OTRS Business Solution™)',
      'NavBar' => 'Reports',
      'Prio' => '100',
      'Type' => ''
    },
    {
      'AccessKey' => '',
      'Block' => '',
      'Description' => '',
      'GroupRo' => [
        'stats'
      ],
      'Link' => 'Action=AgentStatistics;Subaction=Overview',
      'LinkOption' => '',
      'Name' => 'Statistics',
      'NavBar' => 'Reports',
      'Prio' => '200',
      'Type' => ''
    }
  ],
  'NavBarName' => 'Reports',
  'Title' => 'Statistics'
};
```

Framework → Frontend::Agent::NavBarModule

Frontend::NavBarModule###6-CustomerCompany

Frontend module registration (disable company link if no company feature is used).

Default value:

```
$Self->{'Frontend::NavBarModule'}->{'6-CustomerCompany'} =  {
  'Module' => 'Kernel::Output::HTML::NavBar::CustomerCompany'
};
```

Frontend::NavBarModule###7-AgentTicketService

Frontend module registration (disable AgentTicketService link if Ticket Serivice feature is not used).

Default value:

```
$Self->{'Frontend::NavBarModule'}->{'7-AgentTicketService'} =  {
  'Module' => 'Kernel::Output::HTML::NavBar::AgentTicketService'
};
```

Framework → Frontend::Agent::Preferences

PreferencesTable

Defines the name of the table where the user preferences are stored.

This setting can not be deactivated.

Default value:

```
$Self->{'PreferencesTable'} =  'user_preferences';
```

PreferencesTableKey

Defines the column to store the keys for the preferences table.

This setting can not be deactivated.

Default value:

```
$Self->{'PreferencesTableKey'} =  'preferences_key';
```

PreferencesTableValue

Defines the name of the column to store the data in the preferences table.

This setting can not be deactivated.

Default value:

```
$Self->{'PreferencesTableValue'} =  'preferences_value';
```

PreferencesTableUserID

Defines the name of the column to store the user identifier in the preferences table.

This setting can not be deactivated.

Default value:

```
$Self->{'PreferencesTableUserID'} =  'user_id';
```

PreferencesView

Sets the display order of the different items in the preferences view.

This setting can not be deactivated.

Default value:

```
$Self->{'PreferencesView'} =  [
  'User Profile',
  'Notification Settings',
  'Other Settings'
];
```

PreferencesGroups###Password

Defines the config parameters of this item, to be shown in the preferences view. 'PasswordRegExp' allows to match passwords against a regular expression. Define the minimum number of characters using 'PasswordMinSize'. Define if at least 2 lowercase and 2 uppercase letter characters are needed by setting the appropriate option to '1'. 'PasswordMin2Characters' defines if the password needs to contain at least 2 letter characters (set to 0 or 1). 'PasswordNeedDigit' controls the need of at least 1 digit (set to 0 or 1 to control). 'PasswordMaxLoginFailed' allows to set an agent to invalid-temporarily if max failed logins reached.

Default value:

```
$Self->{'PreferencesGroups'}->{'Password'} =  {
  'Active' => '1',
  'Area' => 'Agent',
  'Column' => 'User Profile',
  'Label' => 'Change password',
  'Module' => 'Kernel::Output::HTML::Preferences::Password',
  'PasswordMaxLoginFailed' => '0',
  'PasswordMin2Characters' => '0',
  'PasswordMin2Lower2UpperCharacters' => '0',
  'PasswordMinSize' => '0',
  'PasswordNeedDigit' => '0',
  'PasswordRegExp' => '',
  'Prio' => '0500'
};
```

PreferencesGroups###GoogleAuthenticatorSecretKey

Defines the config parameters of this item, to be shown in the preferences view.

Default value:

```
$Self->{'PreferencesGroups'}->{'GoogleAuthenticatorSecretKey'} =  {
  'Active' => '0',
```

```
  'Block' => 'Input',
  'Column' => 'User Profile',
  'Desc' => 'Enter your shared secret to enable two factor authentication.',
  'Key' => 'Shared Secret',
  'Label' => 'Google Authenticator',
  'Module' => 'Kernel::Output::HTML::Preferences::Generic',
  'PrefKey' => 'UserGoogleAuthenticatorSecretKey',
  'Prio' => '0600'
};
```

PreferencesGroups###SpellDict

Defines the config parameters of this item, to be shown in the preferences view. Take care to maintain the dictionaries installed in the system in the data section.

This setting is not active by default.

Default value:

```
 $Self->{'PreferencesGroups'}->{'SpellDict'} =  {
  'Active' => '1',
  'Column' => 'User Profile',
  'Data' => {
    'deutsch' => 'Deutsch',
    'english' => 'English'
  },
  'DataSelected' => 'english',
  'Key' => 'Default spelling dictionary',
  'Label' => 'Spelling Dictionary',
  'Module' => 'Kernel::Output::HTML::Preferences::Generic',
  'PrefKey' => 'UserSpellDict',
  'Prio' => '2000'
};
```

PreferencesGroups###Comment

Defines the config parameters of this item, to be shown in the preferences view.

Default value:

```
 $Self->{'PreferencesGroups'}->{'Comment'} =  {
  'Active' => '0',
  'Block' => 'Input',
  'Column' => 'Other Settings',
  'Data' => '[% Env("UserComment") %]',
  'Key' => 'Comment',
  'Label' => 'Comment',
  'Module' => 'Kernel::Output::HTML::Preferences::Generic',
  'PrefKey' => 'UserComment',
  'Prio' => '6000'
};
```

PreferencesGroups###Language

Defines the config parameters of this item, to be shown in the preferences view.

Default value:

```
 $Self->{'PreferencesGroups'}->{'Language'} =  {
  'Active' => '1',
  'Column' => 'User Profile',
  'Key' => 'Language',
  'Label' => 'Language',
  'Module' => 'Kernel::Output::HTML::Preferences::Language',
  'PrefKey' => 'UserLanguage',
  'Prio' => '1000'
};
```

PreferencesGroups###Skin

Defines the config parameters of this item, to be shown in the preferences view.

Default value:

```
$Self->{'PreferencesGroups'}->{'Skin'} =  {
  'Active' => '1',
  'Column' => 'Other Settings',
  'Key' => 'Skin',
  'Label' => 'Skin',
  'Module' => 'Kernel::Output::HTML::Preferences::Skin',
  'PrefKey' => 'UserSkin',
  'Prio' => '100'
};
```

PreferencesGroups###Theme

Defines the config parameters of this item, to be shown in the preferences view.

Default value:

```
$Self->{'PreferencesGroups'}->{'Theme'} =  {
  'Active' => '1',
  'Column' => 'User Profile',
  'Key' => 'Frontend theme',
  'Label' => 'Theme',
  'Module' => 'Kernel::Output::HTML::Preferences::Theme',
  'PrefKey' => 'UserTheme',
  'Prio' => '3000'
};
```

PreferencesGroups###OutOfOffice

Defines the config parameters of this item, to be shown in the preferences view.

Default value:

```
$Self->{'PreferencesGroups'}->{'OutOfOffice'} =  {
  'Active' => '1',
  'Block' => 'OutOfOffice',
  'Column' => 'User Profile',
  'Key' => '',
  'Label' => 'Out Of Office Time',
  'Module' => 'Kernel::Output::HTML::Preferences::OutOfOffice',
  'PrefKey' => 'UserOutOfOffice',
  'Prio' => '4000'
};
```

PreferencesGroups###TimeZone

Defines the config parameters of this item, to be shown in the preferences view.

Default value:

```
$Self->{'PreferencesGroups'}->{'TimeZone'} =  {
  'Active' => '1',
  'Column' => 'User Profile',
  'Key' => 'Time Zone',
  'Label' => 'Time Zone',
  'Module' => 'Kernel::Output::HTML::Preferences::TimeZone',
  'PrefKey' => 'UserTimeZone',
  'Prio' => '5000'
};
```

PreferencesGroups###CSVSeparator

Gives end users the possibility to override the separator character for CSV files, defined in the translation files.

This setting is not active by default.

Default value:

```
$Self->{'PreferencesGroups'}->{'CSVSeparator'} =  {
  'Active' => '1',
  'Column' => 'Other Settings',
  'Data' => {
    '' => '',
```

```
    ',' => ',',
    ';' => ';',
    '\\t' => 'tab',
    '|' => '|'
  },
  'DataSelected' => '0',
  'Desc' => 'Select the separator character used in CSV files (stats and searches). If
 you don\'t select a separator here, the default separator for your language will be
 used.',
  'Key' => 'CSV Separator',
  'Label' => 'CSV Separator',
  'Module' => 'Kernel::Output::HTML::Preferences::Generic',
  'PrefKey' => 'UserCSVSeparator',
  'Prio' => '4000'
};
```

Framework → Frontend::Agent::SearchRouter

Frontend::SearchDefault

Search backend default router.

Default value:

```
$Self->{'Frontend::SearchDefault'} =  'Action=AgentTicketSearch;Subaction=AJAX';
```

Framework → Frontend::Agent::Stats

Stats::SearchPageShown

Defines the default maximum number of statistics per page on the overview screen.

This setting can not be deactivated.

Default value:

```
$Self->{'Stats::SearchPageShown'} =  '50';
```

Stats::DefaultSelectedDynamicObject

Defines the default selection at the drop down menu for dynamic objects (Form: Common Specification).

This setting can not be deactivated.

Default value:

```
$Self->{'Stats::DefaultSelectedDynamicObject'} =  'Ticket';
```

Stats::DefaultSelectedPermissions

Defines the default selection at the drop down menu for permissions (Form: Common Specification).

This setting can not be deactivated.

Default value:

```
$Self->{'Stats::DefaultSelectedPermissions'} =  [
  'stats'
];
```

Stats::DefaultSelectedFormat

Defines the default selection at the drop down menu for stats format (Form: Common Specification). Please insert the format key (see Stats::Format).

This setting can not be deactivated.

Default value:

```
$Self->{'Stats::DefaultSelectedFormat'} =  [
  'Print',
  'CSV',
  'Excel',
  'D3::BarChart',
  'D3::LineChart',
  'D3::StackedAreaChart'
];
```

Stats::SearchLimit
Defines the search limit for the stats.

This setting can not be deactivated.

Default value:

```
$Self->{'Stats::SearchLimit'} =  '1000';
```

Stats::Format
Defines all the possible stats output formats.

This setting can not be deactivated.

Default value:

```
$Self->{'Stats::Format'} =  {
  'CSV' => 'CSV',
  'D3::BarChart' => 'Graph: Bar Chart',
  'D3::LineChart' => 'Graph: Line Chart',
  'D3::StackedAreaChart' => 'Graph: Stacked Area Chart',
  'Excel' => 'Excel',
  'Print' => 'Print'
};
```

Stats::ExchangeAxis
Allows agents to exchange the axis of a stat if they generate one.

This setting can not be deactivated.

Default value:

```
$Self->{'Stats::ExchangeAxis'} =  '0';
```

Stats::UseAgentElementInStats
Allows agents to generate individual-related stats.

This setting can not be deactivated.

Default value:

```
$Self->{'Stats::UseAgentElementInStats'} =  '0';
```

Stats::UseInvalidAgentInStats
Allows invalid agents to generate individual-related stats.

This setting can not be deactivated.

Default value:

```
$Self->{'Stats::UseInvalidAgentInStats'} =  '1';
```

Stats::CustomerIDAsMultiSelect
Shows all the customer identifiers in a multi-select field (not useful if you have a lot of customer identifiers).

This setting can not be deactivated.

Default value:

```
$Self->{'Stats::CustomerIDAsMultiSelect'} =  '1';
```

Framework → Frontend::Customer

CustomerHeadline

The headline shown in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerHeadline'} =  'Example Company';
```

CustomerLogo

The logo shown in the header of the customer interface. The URL to the image can be a relative URL to the skin image directory, or a full URL to a remote web server.

This setting is not active by default.

Default value:

```
$Self->{'CustomerLogo'} =  {
  'StyleHeight' => '50px',
  'StyleRight' => '25px',
  'StyleTop' => '2px',
  'StyleWidth' => '135px',
  'URL' => 'skins/Customer/default/img/logo.png'
};
```

CustomerPanelUserID

Defines the user identifier for the customer panel.

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerPanelUserID'} =  '1';
```

CustomerGroupSupport

Activates support for customer groups.

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerGroupSupport'} =  '0';
```

CustomerGroupAlwaysGroups

Defines the groups every customer user will be in (if CustomerGroupSupport is enabled and you don't want to manage every user for these groups).

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerGroupAlwaysGroups'} =  [
  'users'
];
```

CustomerPanelLoginURL

Defines an alternate login URL for the customer panel..

This setting is not active by default.

Default value:

```
$Self->{'CustomerPanelLoginURL'} =  'http://host.example.com/cgi-bin/login.pl';
```

CustomerPanelLogoutURL

Defines an alternate logout URL for the customer panel.

This setting is not active by default.

Default value:

```
$Self->{'CustomerPanelLogoutURL'} =  'http://host.example.com/cgi-bin/login.pl';
```

Frontend::CustomerUser::Item###1-GoogleMaps

Defines a customer item, which generates a google maps icon at the end of a customer info block.

Default value:

```
$Self->{'Frontend::CustomerUser::Item'}->{'1-GoogleMaps'} =  {
  'Attributes' => 'UserStreet;UserCity;UserCountry;',
  'CSS' => 'Core.Agent.CustomerUser.GoogleMaps.css',
  'CSSClass' => 'GoogleMaps',
  'IconName' => 'fa-globe',
  'Module' => 'Kernel::Output::HTML::CustomerUser::Generic',
  'Required' => 'UserStreet;UserCity;',
  'Target' => '_blank',
  'Text' => 'Location',
  'URL' => 'http://maps.google.com/maps?z=7&q='
};
```

Frontend::CustomerUser::Item###2-Google

Defines a customer item, which generates a google icon at the end of a customer info block.

This setting is not active by default.

Default value:

```
$Self->{'Frontend::CustomerUser::Item'}->{'2-Google'} =  {
  'Attributes' => 'UserFirstname;UserLastname;',
  'CSS' => 'Core.Agent.CustomerUser.Google.css',
  'CSSClass' => 'Google',
  'IconName' => 'fa-google',
  'Module' => 'Kernel::Output::HTML::CustomerUser::Generic',
  'Required' => 'UserFirstname;UserLastname;',
  'Target' => '_blank',
  'Text' => 'Google',
  'URL' => 'http://google.com/search?q='
};
```

Frontend::CustomerUser::Item###2-LinkedIn

Defines a customer item, which generates a LinkedIn icon at the end of a customer info block.

This setting is not active by default.

Default value:

```
$Self->{'Frontend::CustomerUser::Item'}->{'2-LinkedIn'} =  {
  'Attributes' => 'UserFirstname;UserLastname;',
  'CSS' => 'Core.Agent.CustomerUser.LinkedIn.css',
  'CSSClass' => 'LinkedIn',
  'IconName' => 'fa-linkedin',
  'Module' => 'Kernel::Output::HTML::CustomerUser::Generic',
  'Required' => 'UserFirstname;UserLastname;',
  'Target' => '_blank',
  'Text' => 'LinkedIn',
```

```
  'URL' => 'http://www.linkedin.com/commonSearch?type=people&keywords='
};
```

Frontend::CustomerUser::Item###3-XING

Defines a customer item, which generates a XING icon at the end of a customer info block.

This setting is not active by default.

Default value:

```
$Self->{'Frontend::CustomerUser::Item'}->{'3-XING'} =  {
  'Attributes' => 'UserFirstname;UserLastname;',
  'CSS' => 'Core.Agent.CustomerUser.Xing.css',
  'CSSClass' => 'Xing',
  'IconName' => 'fa-xing',
  'Module' => 'Kernel::Output::HTML::CustomerUser::Generic',
  'Required' => 'UserFirstname;UserLastname;',
  'Target' => '_blank',
  'Text' => 'XING',
  'URL' => 'https://www.xing.com/app/search?op=search;keywords='
};
```

CustomerPanelPreApplicationModule###CustomerAccept

This module and its PreRun() function will be executed, if defined, for every request. This module is useful to check some user options or to display news about new applications.

This setting is not active by default.

Default value:

```
$Self->{'CustomerPanelPreApplicationModule'}->{'CustomerAccept'} =
'Kernel::Modules::CustomerAccept';
```

CustomerPanel::InfoKey

Defines the key to check with CustomerAccept. If this user preferences key is true, then the message is accepted by the system.

This setting is not active by default.

Default value:

```
$Self->{'CustomerPanel::InfoKey'} =  'CustomerAccept1';
```

CustomerPanel::InfoFile

Defines the path of the shown info file, that is located under Kernel/Output/HTML/Templates/Standard/CustomerAccept.tt.

This setting is not active by default.

Default value:

```
$Self->{'CustomerPanel::InfoFile'} =  'CustomerAccept';
```

CustomerPanelLostPassword

Activates lost password feature for customers.

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerPanelLostPassword'} =  '1';
```

CustomerPanelCreateAccount

Enables customers to create their own accounts.

This setting can not be deactivated.

Default value:

```
 $Self->{'CustomerPanelCreateAccount'} =  '1';
```

CustomerPanelCreateAccount::MailRestrictions::Whitelist
If active, one of the regular expressions has to match the user's email address to allow registration.

This setting is not active by default.

Default value:

```
 $Self->{'CustomerPanelCreateAccount::MailRestrictions::Whitelist'} =  [
  '\\@your\\.domain\\.example$'
];
```

CustomerPanelCreateAccount::MailRestrictions::Blacklist
If active, none of the regular expressions may match the user's email address to allow registration.

This setting is not active by default.

Default value:

```
 $Self->{'CustomerPanelCreateAccount::MailRestrictions::Blacklist'} =  [
  '\\@your\\.domain\\.example$'
];
```

CustomerPanelSubjectLostPasswordToken
Defines the subject for notification mails sent to customers, with token about new requested password.

This setting can not be deactivated.

Default value:

```
 $Self->{'CustomerPanelSubjectLostPasswordToken'} =  'New OTRS password request';
```

CustomerPanelBodyLostPasswordToken
Defines the body text for notification mails sent to customers, with token about new requested password (after using this link the new password will be sent).

This setting can not be deactivated.

Default value:

```
 $Self->{'CustomerPanelBodyLostPasswordToken'} =  'Hi <OTRS_USERFIRSTNAME>,

You or someone impersonating you has requested to change your OTRS
password.

If you want to do this, click on this link. You will receive another email containing
 the password.

<OTRS_CONFIG_HttpType>://<OTRS_CONFIG_FQDN>/<OTRS_CONFIG_ScriptAlias>customer.pl?
Action=CustomerLostPassword;Token=<OTRS_TOKEN>

If you did not request a new password, please ignore this email.
            ';
```

CustomerPanelSubjectLostPassword
Defines the subject for notification mails sent to customers, about new password.

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerPanelSubjectLostPassword'} =  'New OTRS password';
```

CustomerPanelBodyLostPassword

Defines the body text for notification mails sent to customers, about new password (after using this link the new password will be sent).

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerPanelBodyLostPassword'} =  'Hi <OTRS_USERFIRSTNAME>,


New password: <OTRS_NEWPW>

<OTRS_CONFIG_HttpType>://<OTRS_CONFIG_FQDN>/<OTRS_CONFIG_ScriptAlias>customer.pl
            ';
```

CustomerPanelSubjectNewAccount

Defines the subject for notification mails sent to customers, about new account.

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerPanelSubjectNewAccount'} =  'New OTRS Account!';
```

CustomerPanelBodyNewAccount

Defines the body text for notification mails sent to customers, about new account.

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerPanelBodyNewAccount'} =  'Hi <OTRS_USERFIRSTNAME>,

You or someone impersonating you has created a new OTRS account for
you.

Full name: <OTRS_USERFIRSTNAME> <OTRS_USERLASTNAME>
User name: <OTRS_USERLOGIN>
Password : <OTRS_USERPASSWORD>

You can log in via the following URL. We encourage you to change your password
via the Preferences button after logging in.

<OTRS_CONFIG_HttpType>://<OTRS_CONFIG_FQDN>/<OTRS_CONFIG_ScriptAlias>customer.pl
            ';
```

Loader::Customer::Skin###000-default

Default skin for the customer interface.

Default value:

```
$Self->{'Loader::Customer::Skin'}->{'000-default'} =  {
  'Description' => 'This is the default orange - black skin for the customer
 interface.',
  'HomePage' => 'www.otrs.org',
  'InternalName' => 'default',
  'VisibleName' => 'Default'
};
```

Loader::Customer::SelectedSkin

The customer skin's InternalName which should be used in the customer interface. Please check the available skins in Frontend::Customer::Skins.

This setting can not be deactivated.

Default value:

```
$Self->{'Loader::Customer::SelectedSkin'} =  'default';
```

Loader::Customer::SelectedSkin::HostBased

It is possible to configure different skins, for example to distinguish between diferent customers, to be used on a per-domain basis within the application. Using a regular expression (regex), you can configure a Key/Content pair to match a domain. The value in "Key" should match the domain, and the value in "Content" should be a valid skin on your system. Please see the example entries for the proper form of the regex.

This setting is not active by default.

Default value:

```
$Self->{'Loader::Customer::SelectedSkin::HostBased'} =  {
  'host1\\.example\\.com' => 'Someskin1',
  'host2\\.example\\.com' => 'Someskin2'
};
```

AutoComplete::Customer###Default

Defines the config options for the autocompletion feature.

Default value:

```
$Self->{'AutoComplete::Customer'}->{'Default'} =  {
  'AutoCompleteActive' => '1',
  'ButtonText' => 'Search',
  'MaxResultsDisplayed' => '20',
  'MinQueryLength' => '2',
  'QueryDelay' => '100'
};
```

ModernizeCustomerFormFields

Use new type of select and autocomplete fields in customer interface, where applicable (InputFields).

This setting can not be deactivated.

Default value:

```
$Self->{'ModernizeCustomerFormFields'} =  '1';
```

Framework → Frontend::Customer::Auth

Customer::AuthModule

Defines the module to authenticate customers.

This setting can not be deactivated.

Default value:

```
$Self->{'Customer::AuthModule'} =  'Kernel::System::CustomerAuth::DB';
```

Customer::AuthModule::DB::CryptType

If "DB" was selected for Customer::AuthModule, the crypt type of passwords must be specified.

This setting can not be deactivated.

Default value:

```
$Self->{'Customer::AuthModule::DB::CryptType'} =  'sha2';
```

Customer::AuthModule::DB::Table
If "DB" was selected for Customer::AuthModule, the name of the table where your customer data should be stored must be specified.

This setting can not be deactivated.

Default value:

```
$Self->{'Customer::AuthModule::DB::Table'} =  'customer_user';
```

Customer::AuthModule::DB::CustomerKey
If "DB" was selected for Customer::AuthModule, the name of the column for the CustomerKey in the customer table must be specified.

This setting can not be deactivated.

Default value:

```
$Self->{'Customer::AuthModule::DB::CustomerKey'} =  'login';
```

Customer::AuthModule::DB::CustomerPassword
If "DB" was selected for Customer::AuthModule, the column name for the CustomerPassword in the customer table must be specified.

This setting can not be deactivated.

Default value:

```
$Self->{'Customer::AuthModule::DB::CustomerPassword'} =  'pw';
```

Customer::AuthModule::DB::DSN
If "DB" was selected for Customer::AuthModule, the DSN for the connection to the customer table must be specified.

This setting is not active by default.

Default value:

```
$Self->{'Customer::AuthModule::DB::DSN'} =
'DBI:mysql:database=customerdb;host=customerdbhost';
```

Customer::AuthModule::DB::User
If "DB" was selected for Customer::AuthModule, a username to connect to the customer table can be specified.

This setting is not active by default.

Default value:

```
$Self->{'Customer::AuthModule::DB::User'} =  'some_user';
```

Customer::AuthModule::DB::Password
If "DB" was selected for Customer::AuthModule, a password to connect to the customer table can be specified.

This setting is not active by default.

Default value:

```
$Self->{'Customer::AuthModule::DB::Password'} =  'some_password';
```

Customer::AuthModule::DB::Type
If "DB" was selected for Customer::AuthModule, a database driver (normally autodetection is used) can be specified.

This setting is not active by default.

Default value:

```
$Self->{'Customer::AuthModule::DB::Type'} =  'mysql';
```

Customer::AuthModule::HTTPBasicAuth::Replace

If "HTTPBasicAuth" was selected for Customer::AuthModule, you can specify to strip leading parts of user names (e. g. for domains like example_domain\user to user).

This setting is not active by default.

Default value:

```
$Self->{'Customer::AuthModule::HTTPBasicAuth::Replace'} =  'example_domain\\\\';
```

Customer::AuthModule::HTTPBasicAuth::ReplaceRegExp

If "HTTPBasicAuth" was selected for Customer::AuthModule, you can specify (by using a RegExp) to strip parts of REMOTE_USER (e. g. for to remove trailing domains). RegExp-Note, $1 will be the new Login.

This setting is not active by default.

Default value:

```
$Self->{'Customer::AuthModule::HTTPBasicAuth::ReplaceRegExp'} =  '^(.+?)@.+?$';
```

Customer::AuthModule::LDAP::Host

If "LDAP" was selected for Customer::AuthModule, the LDAP host can be specified.

This setting is not active by default.

Default value:

```
$Self->{'Customer::AuthModule::LDAP::Host'} =  'ldap.example.com';
```

Customer::AuthModule::LDAP::BaseDN

If "LDAP" was selected for Customer::AuthModule, the BaseDN must be specified.

This setting is not active by default.

Default value:

```
$Self->{'Customer::AuthModule::LDAP::BaseDN'} =  'dc=example,dc=com';
```

Customer::AuthModule::LDAP::UID

If "LDAP" was selected for Customer::AuthModule, the user identifier must be specified.

This setting is not active by default.

Default value:

```
$Self->{'Customer::AuthModule::LDAP::UID'} =  'uid';
```

Customer::AuthModule::LDAP::GroupDN

If "LDAP" was selected for Customer::Authmodule, you can check if the user is allowed to authenticate because he is in a posixGroup, e.g. user needs to be in a group xyz to use OTRS. Specify the group, who may access the system.

This setting is not active by default.

Default value:

```
$Self->{'Customer::AuthModule::LDAP::GroupDN'} =
'cn=otrsallow,ou=posixGroups,dc=example,dc=com';
```

Customer::AuthModule::LDAP::AccessAttr

If "LDAP" was selected for Customer::AuthModule, you can specify access attributes here.

This setting is not active by default.

Default value:

```
$Self->{'Customer::AuthModule::LDAP::AccessAttr'} =  'memberUid';
```

Customer::AuthModule::LDAP::UserAttr

If "LDAP" was selected for Customer::AuthModule, user attributes can be specified. For LDAP posixGroups use UID, for non LDAP posixGroups use full user DN.

This setting is not active by default.

Default value:

```
$Self->{'Customer::AuthModule::LDAP::UserAttr'} =  'UID';
```

Customer::AuthModule::LDAP::SearchUserDN

If "LDAP" was selected for Customer::AuthModule and your users have only anonymous access to the LDAP tree, but you want to search through the data, you can do this with a user who has access to the LDAP directory. Specify the username for this special user here.

This setting is not active by default.

Default value:

```
$Self->{'Customer::AuthModule::LDAP::SearchUserDN'} =
'cn=binduser,ou=users,dc=example,dc=com';
```

Customer::AuthModule::LDAP::SearchUserPw

If "LDAP" was selected for Customer::AuthModule and your users have only anonymous access to the LDAP tree, but you want to search through the data, you can do this with a user who has access to the LDAP directory. Specify the password for this special user here.

This setting is not active by default.

Default value:

```
$Self->{'Customer::AuthModule::LDAP::SearchUserPw'} =  'some_password';
```

Customer::AuthModule::LDAP::AlwaysFilter

If "LDAP" was selected, you can add a filter to each LDAP query, e.g. (mail=*), (objectclass=user) or (!objectclass=computer).

This setting is not active by default.

Default value:

```
$Self->{'Customer::AuthModule::LDAP::AlwaysFilter'} =  '(!objectclass=computer)';
```

Customer::AuthModule::LDAP::UserSuffix

If "LDAP" was selected for Customer::AuthModule and if you want to add a suffix to every customer login name, specifiy it here, e. g. you just want to write the username user but in your LDAP directory exists user@domain.

This setting is not active by default.

Default value:

```
$Self->{'Customer::AuthModule::LDAP::UserSuffix'} =  '@domain.com';
```

Customer::AuthModule::LDAP::Params
If "LDAP" was selected for Customer::AuthModule and special paramaters are needed for the Net::LDAP perl module, you can specify them here. See "perldoc Net::LDAP" for more information about the parameters.

This setting is not active by default.

Default value:

```
$Self->{'Customer::AuthModule::LDAP::Params'} =  {
  'async' => '0',
  'port' => '389',
  'timeout' => '120',
  'version' => '3'
};
```

Customer::AuthModule::LDAP::Die
If "LDAP" was selected for Customer::AuthModule, you can specify if the applications will stop if e. g. a connection to a server can't be established due to network problems.

Default value:

```
$Self->{'Customer::AuthModule::LDAP::Die'} =  '1';
```

Customer::AuthModule::Radius::Host
If "Radius" was selected for Customer::AuthModule, the radius host must be specified.

This setting is not active by default.

Default value:

```
$Self->{'Customer::AuthModule::Radius::Host'} =  'radiushost';
```

Customer::AuthModule::Radius::Password
If "Radius" was selected for Customer::AuthModule, the password to authenticate to the radius host must be specified.

This setting is not active by default.

Default value:

```
$Self->{'Customer::AuthModule::Radius::Password'} =  'radiussecret';
```

Customer::AuthModule::Radius::Die
If "Radius" was selected for Customer::AuthModule, you can specify if the applications will stop if e. g. a connection to a server can't be established due to network problems.

Default value:

```
$Self->{'Customer::AuthModule::Radius::Die'} =  '1';
```

Framework → Frontend::Customer::Auth::TwoFactor

Customer::AuthTwoFactorModule
Defines the two-factor module to authenticate customers.

This setting is not active by default.

Default value:

```
$Self->{'Customer::AuthTwoFactorModule'} =
'Kernel::System::CustomerAuth::TwoFactor::GoogleAuthenticator';
```

Customer::AuthTwoFactorModule::SecretPreferencesKey
Defines the customer preferences key where the shared secret key is stored.

This setting can not be deactivated.

Default value:

```
$Self->{'Customer::AuthTwoFactorModule::SecretPreferencesKey'} =
'UserGoogleAuthenticatorSecretKey';
```

Customer::AuthTwoFactorModule::AllowEmptySecret
Defines if customers should be allowed to login if they have no shared secret stored in their preferences and therefore are not using two-factor authentication.

Default value:

```
$Self->{'Customer::AuthTwoFactorModule::AllowEmptySecret'} =  '1';
```

Customer::AuthTwoFactorModule::AllowPreviousToken
Defines if the previously valid token should be accepted for authentication. This is slightly less secure but gives users 30 seconds more time to enter their one-time password.

Default value:

```
$Self->{'Customer::AuthTwoFactorModule::AllowPreviousToken'} =  '1';
```

Framework → Frontend::Customer::ModuleMetaHead

CustomerFrontend::HeaderMetaModule###1-Refresh
Defines the module to generate code for periodic page reloads.

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerFrontend::HeaderMetaModule'}->{'1-Refresh'} =  {
  'Module' => 'Kernel::Output::HTML::HeaderMeta::Refresh'
};
```

Framework → Frontend::Customer::ModuleNotify

CustomerFrontend::NotifyModule###1-OTRSBusiness
Defines the module to display a notification in different interfaces on different occasions for OTRS Business Solution™.

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerFrontend::NotifyModule'}->{'1-OTRSBusiness'} =  {
  'Module' => 'Kernel::Output::HTML::Notification::CustomerOTRSBusiness'
};
```

CustomerFrontend::NotifyModule###1-ShowAgentOnline
Defines the module that shows the currently logged in agents in the customer interface.

This setting is not active by default.

Default value:

```
$Self->{'CustomerFrontend::NotifyModule'}->{'1-ShowAgentOnline'} =  {
  'IdleMinutes' => '60',
  'Module' => 'Kernel::Output::HTML::Notification::AgentOnline',
```

```
  'ShowEmail' => '1'
};
```

CustomerFrontend::NotifyModule###1-ShowCustomerOnline
Defines the module that shows the currently logged in customers in the customer interface.

This setting is not active by default.

Default value:

```
$Self->{'CustomerFrontend::NotifyModule'}->{'1-ShowCustomerOnline'} =  {
  'Module' => 'Kernel::Output::HTML::Notification::CustomerOnline',
  'ShowEmail' => '1'
};
```

CustomerFrontend::NotifyModule###6-CustomerSystemMaintenance-Check
Defines the module to display a notification in the agent interface, if the agent is logged in while having system maintenance active.

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerFrontend::NotifyModule'}->{'6-CustomerSystemMaintenance-Check'} =  {
  'Module' => 'Kernel::Output::HTML::Notification::CustomerSystemMaintenanceCheck'
};
```

Framework → Frontend::Customer::ModuleRegistration

CustomerFrontend::Module###Logout
Frontend module registration for the customer interface.

Default value:

```
$Self->{'CustomerFrontend::Module'}->{'Logout'} =  {
  'Description' => 'Logout of customer panel.',
  'NavBarName' => '',
  'Title' => ''
};
```

CustomerFrontend::Module###CustomerPreferences
Frontend module registration for the customer interface.

Default value:

```
$Self->{'CustomerFrontend::Module'}->{'CustomerPreferences'} =  {
  'Description' => 'Customer preferences.',
  'NavBarName' => '',
  'Title' => 'Preferences'
};
```

CustomerFrontend::Module###CustomerAccept
Frontend module registration for the customer interface.

Default value:

```
$Self->{'CustomerFrontend::Module'}->{'CustomerAccept'} =  {
  'Description' => 'To accept login information, such as an EULA or license.',
  'NavBarName' => '',
  'Title' => 'Info'
};
```

CustomerFrontend::Module###PictureUpload
Frontend module registration for the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerFrontend::Module'}->{'PictureUpload'} =  {
  'Description' => 'Picture upload module.',
  'NavBarName' => 'Ticket',
  'Title' => 'Picture-Upload'
};
```

Framework → Frontend::Customer::Preferences

CustomerPreferences

Defines the parameters for the customer preferences table.

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerPreferences'} =  {
  'Module' => 'Kernel::System::CustomerUser::Preferences::DB',
  'Params' => {
    'Table' => 'customer_preferences',
    'TableKey' => 'preferences_key',
    'TableUserID' => 'user_id',
    'TableValue' => 'preferences_value'
  }
};
```

CustomerPreferencesView

Sets the order of the different items in the customer preferences view.

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerPreferencesView'} =  [
  'User Profile',
  'Other Settings'
];
```

CustomerPreferencesGroups###Password

Defines all the parameters for this item in the customer preferences. 'PasswordRegExp' allows to match passwords against a regular expression. Define the minimum number of characters using 'PasswordMinSize'. Define if at least 2 lowercase and 2 uppercase letter characters are needed by setting the appropriate option to '1'. 'PasswordMin2Characters' defines if the password needs to contain at least 2 letter characters (set to 0 or 1). 'PasswordNeedDigit' controls the need of at least 1 digit (set to 0 or 1 to control).

Default value:

```
$Self->{'CustomerPreferencesGroups'}->{'Password'} =  {
  'Active' => '1',
  'Area' => 'Customer',
  'Column' => 'Other Settings',
  'Label' => 'Change password',
  'Module' => 'Kernel::Output::HTML::Preferences::Password',
  'PasswordMin2Characters' => '0',
  'PasswordMin2Lower2UpperCharacters' => '0',
  'PasswordMinSize' => '0',
  'PasswordNeedDigit' => '0',
  'PasswordRegExp' => '',
  'Prio' => '1000'
};
```

CustomerPreferencesGroups###GoogleAuthenticatorSecretKey

Defines the config parameters of this item, to be shown in the preferences view.

Default value:

```
$Self->{'CustomerPreferencesGroups'}->{'GoogleAuthenticatorSecretKey'} =  {
  'Active' => '0',
  'Block' => 'Input',
  'Column' => 'Other Settings',
  'Key' => 'Shared Secret',
  'Label' => 'Google Authenticator',
  'Module' => 'Kernel::Output::HTML::Preferences::Generic',
  'PrefKey' => 'UserGoogleAuthenticatorSecretKey',
  'Prio' => '1100'
};
```

CustomerPreferencesGroups###Language

Defines all the parameters for this item in the customer preferences.

Default value:

```
$Self->{'CustomerPreferencesGroups'}->{'Language'} =  {
  'Active' => '1',
  'Column' => 'User Profile',
  'Key' => 'Language',
  'Label' => 'Interface language',
  'Module' => 'Kernel::Output::HTML::Preferences::Language',
  'PrefKey' => 'UserLanguage',
  'Prio' => '2000'
};
```

CustomerPreferencesGroups###Theme

Defines all the parameters for this item in the customer preferences.

Default value:

```
$Self->{'CustomerPreferencesGroups'}->{'Theme'} =  {
  'Active' => '0',
  'Column' => 'User Profile',
  'Key' => 'Select your frontend Theme.',
  'Label' => 'Theme',
  'Module' => 'Kernel::Output::HTML::Preferences::Theme',
  'PrefKey' => 'UserTheme',
  'Prio' => '1000'
};
```

CustomerPreferencesGroups###TimeZone

Defines all the parameters for this item in the customer preferences.

Default value:

```
$Self->{'CustomerPreferencesGroups'}->{'TimeZone'} =  {
  'Active' => '1',
  'Column' => 'User Profile',
  'Key' => 'Time Zone',
  'Label' => 'Time Zone',
  'Module' => 'Kernel::Output::HTML::Preferences::TimeZone',
  'PrefKey' => 'UserTimeZone',
  'Prio' => '5000'
};
```

CustomerPreferencesGroups###PGP

Defines all the parameters for this item in the customer preferences.

Default value:

```
$Self->{'CustomerPreferencesGroups'}->{'PGP'} =  {
  'Active' => '1',
  'Column' => 'Other Settings',
  'Key' => 'PGP Key Upload',
  'Label' => 'PGP Key',
  'Module' => 'Kernel::Output::HTML::Preferences::PGP',
```

```
  'PrefKey' => 'UserPGPKey',
  'Prio' => '10000'
};
```

CustomerPreferencesGroups###SMIME
Defines all the parameters for this item in the customer preferences.

Default value:

```
 $Self->{'CustomerPreferencesGroups'}->{'SMIME'} =  {
  'Active' => '1',
  'Column' => 'Other Settings',
  'Key' => 'S/MIME Certificate Upload',
  'Label' => 'S/MIME Certificate',
  'Module' => 'Kernel::Output::HTML::Preferences::SMIME',
  'PrefKey' => 'UserSMIMEKey',
  'Prio' => '11000'
};
```

Framework → Frontend::Public

PublicFrontend::CommonParam###Action
Defines the default value for the action parameter for the public frontend. The action parameter is used in the scripts of the system.

This setting can not be deactivated.

Default value:

```
 $Self->{'PublicFrontend::CommonParam'}->{'Action'} =  'PublicDefault';
```

Framework → Frontend::Public::ModuleRegistration

PublicFrontend::Module###PublicDefault
Frontend module registration for the agent interface.

Default value:

```
 $Self->{'PublicFrontend::Module'}->{'PublicDefault'} =  {
  'Description' => 'PublicDefault',
  'NavBarName' => '',
  'Title' => 'PublicDefault'
};
```

PublicFrontend::Module###PublicRepository
Frontend module registration for the agent interface.

Default value:

```
 $Self->{'PublicFrontend::Module'}->{'PublicRepository'} =  {
  'Description' => 'PublicRepository',
  'NavBarName' => '',
  'Title' => 'PublicRepository'
};
```

PublicFrontend::Module###PublicSupportDataCollector
Frontend module registration for the agent interface.

Default value:

```
 $Self->{'PublicFrontend::Module'}->{'PublicSupportDataCollector'} =  {
  'Description' => 'PublicSupportDataCollector',
  'NavBarName' => '',
  'Title' => 'PublicSupportDataCollector'
};
```

Framework → SystemMaintenance

SystemMaintenance::TimeNotifyUpcomingMaintenance
Sets the minutes a notification is shown for notice about upcoming system maintenance period.

Default value:

```
$Self->{'SystemMaintenance::TimeNotifyUpcomingMaintenance'} =  '30';
```

SystemMaintenance::IsActiveDefaultNotification
Sets the default message for the notification is shown on a running system maintenance period.

Default value:

```
$Self->{'SystemMaintenance::IsActiveDefaultNotification'} =  'We are performing
scheduled maintenance.';
```

SystemMaintenance::IsActiveDefaultLoginMessage
Sets the default message for the login screen on Agent and Customer interface, it's shown when a running system maintenance period is active.

Default value:

```
$Self->{'SystemMaintenance::IsActiveDefaultLoginMessage'} =  'We are performing
scheduled maintenance. We should be back online shortly.';
```

SystemMaintenance::IsActiveDefaultLoginErrorMessage
Sets the default error message for the login screen on Agent and Customer interface, it's shown when a running system maintenance period is active.

Default value:

```
$Self->{'SystemMaintenance::IsActiveDefaultLoginErrorMessage'} =  'We are performing
scheduled maintenance. Login is temporarily not available.';
```

5. GenericInterface

GenericInterface → Core::CustomerCompany

CustomerCompany::EventModulePost###1000-GenericInterface
Performs the configured action for each event (as an Invoker) for each configured Webservice.

Default value:

```
$Self->{'CustomerCompany::EventModulePost'}->{'1000-GenericInterface'} =  {
  'Event' => '',
  'Module' => 'Kernel::GenericInterface::Event::Handler',
  'Transaction' => '1'
};
```

GenericInterface → Core::CustomerUser

CustomerUser::EventModulePost###1000-GenericInterface
Performs the configured action for each event (as an Invoker) for each configured Webservice.

Default value:

```
$Self->{'CustomerUser::EventModulePost'}->{'1000-GenericInterface'} =  {
  'Event' => '',
```

```
  'Module' => 'Kernel::GenericInterface::Event::Handler',
  'Transaction' => '1'
};
```

GenericInterface → Core::DynamicField

DynamicField::EventModulePost###1000-GenericInterface
Performs the configured action for each event (as an Invoker) for each configured Webservice.

Default value:

```
$Self->{'DynamicField::EventModulePost'}->{'1000-GenericInterface'} =  {
  'Event' => '',
  'Module' => 'Kernel::GenericInterface::Event::Handler',
  'Transaction' => '1'
};
```

GenericInterface → Core::Package

Package::EventModulePost###1000-GenericInterface
Performs the configured action for each event (as an Invoker) for each configured Webservice.

Default value:

```
$Self->{'Package::EventModulePost'}->{'1000-GenericInterface'} =  {
  'Event' => '',
  'Module' => 'Kernel::GenericInterface::Event::Handler',
  'Transaction' => '1'
};
```

GenericInterface → Core::Queue

Queue::EventModulePost###1000-GenericInterface
Performs the configured action for each event (as an Invoker) for each configured Webservice.

Default value:

```
$Self->{'Queue::EventModulePost'}->{'1000-GenericInterface'} =  {
  'Event' => '',
  'Module' => 'Kernel::GenericInterface::Event::Handler',
  'Transaction' => '1'
};
```

GenericInterface → Core::Ticket

Ticket::EventModulePost###999-GenericInterface
Performs the configured action for each event (as an Invoker) for each configured Webservice.

Default value:

```
$Self->{'Ticket::EventModulePost'}->{'999-GenericInterface'} =  {
  'Event' => '',
  'Module' => 'Kernel::GenericInterface::Event::Handler',
  'Transaction' => '1'
};
```

GenericInterface → Frontend::Admin::ModuleRegistration

Frontend::Module###AdminGenericInterfaceDebugger
Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminGenericInterfaceDebugger'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.GenericInterface.css'
    ],
    'JavaScript' => [
      'Core.Agent.Admin.GenericInterfaceDebugger.js'
    ]
  },
  'Title' => 'GenericInterface Debugger GUI'
};
```

Frontend::Module###AdminGenericInterfaceWebservice

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminGenericInterfaceWebservice'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.GenericInterface.css'
    ],
    'JavaScript' => [
      'Core.Agent.Admin.GenericInterfaceWebservice.js'
    ]
  },
  'NavBarModule' => {
    'Block' => 'System',
    'Description' => 'Create and manage web services.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Web Services',
    'Prio' => '1000'
  },
  'NavBarName' => 'Admin',
  'Title' => 'GenericInterface Web Service GUI'
};
```

Frontend::Module###AdminGenericInterfaceTransportHTTPSOAP

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminGenericInterfaceTransportHTTPSOAP'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.GenericInterface.css',
      'Core.Agent.SortedTree.css'
    ],
    'JavaScript' => [
      'Core.Agent.SortedTree.js'
    ]
  },
  'Title' => 'GenericInterface TransportHTTPSOAP GUI'
};
```

Frontend::Module###AdminGenericInterfaceTransportHTTPREST

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminGenericInterfaceTransportHTTPREST'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.GenericInterface.css'
    ]
  },
  'Title' => 'GenericInterface TransportHTTPREST GUI'
};
```

Frontend::Module###AdminGenericInterfaceWebserviceHistory

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminGenericInterfaceWebserviceHistory'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.GenericInterface.css'
    ],
    'JavaScript' => [
      'Core.Agent.Admin.GenericInterfaceWebserviceHistory.js'
    ]
  },
  'Title' => 'GenericInterface Webservice History GUI'
};
```

Frontend::Module###AdminGenericInterfaceOperationDefault

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminGenericInterfaceOperationDefault'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.GenericInterface.css'
    ],
    'JavaScript' => [
      'Core.Agent.Admin.GenericInterfaceOperation.js'
    ]
  },
  'Title' => 'GenericInterface Operation GUI'
};
```

Frontend::Module###AdminGenericInterfaceInvokerDefault

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminGenericInterfaceInvokerDefault'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.GenericInterface.css'
```

```
    ],
    'JavaScript' => [
      'Core.Agent.Admin.GenericInterfaceInvoker.js'
    ]
  },
  'Title' => 'GenericInterface Invoker GUI'
};
```

Frontend::Module###AdminGenericInterfaceMappingSimple
Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminGenericInterfaceMappingSimple'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.GenericInterface.css'
    ],
    'JavaScript' => [
      'Core.Agent.Admin.GenericInterfaceMappingSimple.js'
    ]
  },
  'Title' => 'GenericInterface Webservice Mapping GUI'
};
```

Frontend::Module###AdminGenericInterfaceMappingXSLT
Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminGenericInterfaceMappingXSLT'} =  {
  'Description' => 'Admin',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.GenericInterface.css'
    ]
  },
  'Title' => 'GenericInterface Webservice Mapping GUI'
};
```

GenericInterface → GenericInterface::Invoker::ModuleRegistration

GenericInterface::Invoker::Module###Test::Test
GenericInterface module registration for the invoker layer.

This setting is not active by default.

Default value:

```
$Self->{'GenericInterface::Invoker::Module'}->{'Test::Test'} =  {
  'ConfigDialog' => 'AdminGenericInterfaceInvokerDefault',
  'Controller' => 'Test',
  'Name' => 'Test'
};
```

GenericInterface::Invoker::Module###Test::TestSimple
GenericInterface module registration for the invoker layer.

This setting is not active by default.

Default value:

```
$Self->{'GenericInterface::Invoker::Module'}->{'Test::TestSimple'} =  {
  'ConfigDialog' => 'AdminGenericInterfaceInvokerDefault',
  'Controller' => 'Test',
  'Name' => 'TestSimple'
};
```

GenericInterface → GenericInterface::Mapping::ModuleRegistration

GenericInterface::Mapping::Module###Test

GenericInterface module registration for the mapping layer.

This setting is not active by default.

Default value:

```
$Self->{'GenericInterface::Mapping::Module'}->{'Test'} =  {
  'ConfigDialog' => ''
};
```

GenericInterface::Mapping::Module###Simple

GenericInterface module registration for the mapping layer.

Default value:

```
$Self->{'GenericInterface::Mapping::Module'}->{'Simple'} =  {
  'ConfigDialog' => 'AdminGenericInterfaceMappingSimple'
};
```

GenericInterface::Mapping::Module###XSLT

GenericInterface module registration for the mapping layer.

Default value:

```
$Self->{'GenericInterface::Mapping::Module'}->{'XSLT'} =  {
  'ConfigDialog' => 'AdminGenericInterfaceMappingXSLT'
};
```

GenericInterface → GenericInterface::Operation::ModuleRegistration

GenericInterface::Operation::Module###Test::Test

GenericInterface module registration for the operation layer.

This setting is not active by default.

Default value:

```
$Self->{'GenericInterface::Operation::Module'}->{'Test::Test'} =  {
  'ConfigDialog' => 'AdminGenericInterfaceOperationDefault',
  'Controller' => 'Test',
  'Name' => 'Test'
};
```

GenericInterface::Operation::Module###Session::SessionCreate

GenericInterface module registration for the operation layer.

Default value:

```
$Self->{'GenericInterface::Operation::Module'}->{'Session::SessionCreate'} =  {
  'ConfigDialog' => 'AdminGenericInterfaceOperationDefault',
  'Controller' => 'Session',
  'Name' => 'SessionCreate'
```

```
};
```

GenericInterface::Operation::Module###Ticket::TicketCreate
GenericInterface module registration for the operation layer.

Default value:

```
$Self->{'GenericInterface::Operation::Module'}->{'Ticket::TicketCreate'} =  {
  'ConfigDialog' => 'AdminGenericInterfaceOperationDefault',
  'Controller' => 'Ticket',
  'Name' => 'TicketCreate'
};
```

GenericInterface::Operation::Module###Ticket::TicketUpdate
GenericInterface module registration for the operation layer.

Default value:

```
$Self->{'GenericInterface::Operation::Module'}->{'Ticket::TicketUpdate'} =  {
  'ConfigDialog' => 'AdminGenericInterfaceOperationDefault',
  'Controller' => 'Ticket',
  'Name' => 'TicketUpdate'
};
```

GenericInterface::Operation::Module###Ticket::TicketGet
GenericInterface module registration for the operation layer.

Default value:

```
$Self->{'GenericInterface::Operation::Module'}->{'Ticket::TicketGet'} =  {
  'ConfigDialog' => 'AdminGenericInterfaceOperationDefault',
  'Controller' => 'Ticket',
  'Name' => 'TicketGet'
};
```

GenericInterface::Operation::Module###Ticket::TicketSearch
GenericInterface module registration for the operation layer.

Default value:

```
$Self->{'GenericInterface::Operation::Module'}->{'Ticket::TicketSearch'} =  {
  'ConfigDialog' => 'AdminGenericInterfaceOperationDefault',
  'Controller' => 'Ticket',
  'Name' => 'TicketGet'
};
```

GenericInterface → GenericInterface::Operation::ResponseLoggingMaxSize

GenericInterface::Operation::ResponseLoggingMaxSize
Defines the maximum size in KiloByte of GenericInterface responses that get logged to the gi_debugger_entry_content table.

This setting can not be deactivated.

Default value:

```
$Self->{'GenericInterface::Operation::ResponseLoggingMaxSize'} =  '200';
```

GenericInterface → GenericInterface::Operation::TicketCreate

GenericInterface::Operation::TicketCreate###ArticleType
Defines the default type of the article for this operation.

This setting can not be deactivated.

Default value:

```
$Self->{'GenericInterface::Operation::TicketCreate'}->{'ArticleType'} =  'webrequest';
```

GenericInterface::Operation::TicketCreate###HistoryType
Defines the history type for this operation, which gets used for ticket history in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'GenericInterface::Operation::TicketCreate'}->{'HistoryType'} =  'NewTicket';
```

GenericInterface::Operation::TicketCreate###HistoryComment
Defines the history comment for this operation, which gets used for ticket history in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'GenericInterface::Operation::TicketCreate'}->{'HistoryComment'} =  '%
%GenericInterface Create';
```

GenericInterface::Operation::TicketCreate###AutoResponseType
Defines the default auto response type of the article for this operation.

This setting can not be deactivated.

Default value:

```
$Self->{'GenericInterface::Operation::TicketCreate'}->{'AutoResponseType'} =  'auto
reply';
```

GenericInterface → GenericInterface::Operation::TicketSearch

GenericInterface::Operation::TicketSearch###SearchLimit
Maximum number of tickets to be displayed in the result of this operation.

This setting can not be deactivated.

Default value:

```
$Self->{'GenericInterface::Operation::TicketSearch'}->{'SearchLimit'} =  '500';
```

GenericInterface::Operation::TicketSearch###SortBy::Default
Defines the default ticket attribute for ticket sorting of the ticket search result of this operation.

This setting can not be deactivated.

Default value:

```
$Self->{'GenericInterface::Operation::TicketSearch'}->{'SortBy::Default'} =  'Age';
```

GenericInterface::Operation::TicketSearch###Order::Default
Defines the default ticket order in the ticket search result of the this operation. Up: oldest on top. Down: latest on top.

This setting can not be deactivated.

Default value:

```
$Self->{'GenericInterface::Operation::TicketSearch'}->{'Order::Default'} =  'Down';
```

GenericInterface → GenericInterface::Operation::TicketUpdate

GenericInterface::Operation::TicketUpdate###ArticleType
Defines the default type of the article for this operation.

This setting can not be deactivated.

Default value:

```
$Self->{'GenericInterface::Operation::TicketUpdate'}->{'ArticleType'} =  'webrequest';
```

GenericInterface::Operation::TicketUpdate###HistoryType
Defines the history type for this operation, which gets used for ticket history in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'GenericInterface::Operation::TicketUpdate'}->{'HistoryType'} =  'AddNote';
```

GenericInterface::Operation::TicketUpdate###HistoryComment
Defines the history comment for this operation, which gets used for ticket history in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'GenericInterface::Operation::TicketUpdate'}->{'HistoryComment'} =  '%
%GenericInterface Note';
```

GenericInterface::Operation::TicketUpdate###AutoResponseType
Defines the default auto response type of the article for this operation.

This setting can not be deactivated.

Default value:

```
$Self->{'GenericInterface::Operation::TicketUpdate'}->{'AutoResponseType'} =  'auto
follow up';
```

GenericInterface → GenericInterface::Transport::ModuleRegistration

GenericInterface::Transport::Module###HTTP::SOAP
GenericInterface module registration for the transport layer.

Default value:

```
$Self->{'GenericInterface::Transport::Module'}->{'HTTP::SOAP'} =  {
  'ConfigDialog' => 'AdminGenericInterfaceTransportHTTPSOAP',
  'Name' => 'SOAP',
  'Protocol' => 'HTTP'
};
```

GenericInterface::Transport::Module###HTTP::REST
GenericInterface module registration for the transport layer.

Default value:

```
$Self->{'GenericInterface::Transport::Module'}->{'HTTP::REST'} =  {
  'ConfigDialog' => 'AdminGenericInterfaceTransportHTTPREST',
  'Name' => 'REST',
  'Protocol' => 'HTTP'
};
```

GenericInterface::Transport::Module###HTTP::Test
GenericInterface module registration for the transport layer.

This setting is not active by default.

Default value:

```
$Self->{'GenericInterface::Transport::Module'}->{'HTTP::Test'} =  {
 'ConfigDialog' => 'AdminGenericInterfaceTransportHTTPTest',
 'Name' => 'Test',
 'Protocol' => 'HTTP'
};
```

GenericInterface → GenericInterface::Webservice

GenericInterface::WebserviceConfig::CacheTTL
Cache time in seconds for the web service config backend.

This setting can not be deactivated.

Default value:

```
$Self->{'GenericInterface::WebserviceConfig::CacheTTL'} =  '86400';
```

GenericInterface::Operation::Common::CachedAuth::AgentCacheTTL
Cache time in seconds for agent authentication in the GenericInterface.

This setting can not be deactivated.

Default value:

```
$Self->{'GenericInterface::Operation::Common::CachedAuth::AgentCacheTTL'} =  '300';
```

GenericInterface::Operation::Common::CachedAuth::CustomerCacheTTL
Cache time in seconds for customer authentication in the GenericInterface.

This setting can not be deactivated.

Default value:

```
$Self->{'GenericInterface::Operation::Common::CachedAuth::CustomerCacheTTL'} =  '300';
```

6. ProcessManagement

ProcessManagement → Core

Process::DynamicFieldProcessManagementProcessID
This option defines the dynamic field in which a Process Management process entity id is stored.

This setting can not be deactivated.

Default value:

```
$Self->{'Process::DynamicFieldProcessManagementProcessID'} =
'ProcessManagementProcessID';
```

Process::DynamicFieldProcessManagementActivityID
This option defines the dynamic field in which a Process Management activity entity id is stored.

This setting can not be deactivated.

Default value:

```
$Self->{'Process::DynamicFieldProcessManagementActivityID'} =
'ProcessManagementActivityID';
```

Process::DefaultQueue
This option defines the process tickets default queue.

This setting can not be deactivated.

Default value:

```
$Self->{'Process::DefaultQueue'} =  'Raw';
```

Process::DefaultState
This option defines the process tickets default state.

This setting can not be deactivated.

Default value:

```
$Self->{'Process::DefaultState'} =  'new';
```

Process::DefaultLock
This option defines the process tickets default lock.

This setting can not be deactivated.

Default value:

```
$Self->{'Process::DefaultLock'} =  'unlock';
```

Process::DefaultPriority
This option defines the process tickets default priority.

This setting can not be deactivated.

Default value:

```
$Self->{'Process::DefaultPriority'} =  '3 normal';
```

Process::Entity::Prefix
Default ProcessManagement entity prefixes for entity IDs that are automatically generated.

This setting can not be deactivated.

Default value:

```
$Self->{'Process::Entity::Prefix'} =  {
  'Activity' => 'A',
  'ActivityDialog' => 'AD',
  'Process' => 'P',
  'Transition' => 'T',
  'TransitionAction' => 'TA'
};
```

Process::CacheTTL
Cache time in seconds for the DB process backend.

This setting can not be deactivated.

Default value:

```
$Self->{'Process::CacheTTL'} =  '3600';
```

Process::NavBarOutput::CacheTTL
Cache time in seconds for the ticket process navigation bar output module.

This setting can not be deactivated.

Default value:

```
$Self->{'Process::NavBarOutput::CacheTTL'} =  '900';
```

ProcessManagement → Core::Ticket

Ticket::EventModulePost###998-TicketProcessTransitions

Event module registration. For more performance you can define a trigger event (e. g. Event => TicketCreate).

Default value:

```
$Self->{'Ticket::EventModulePost'}->{'998-TicketProcessTransitions'} =  {
  'Event' => '',
  'Module' => 'Kernel::System::Ticket::Event::TicketProcessTransitions',
  'Transaction' => '1'
};
```

ProcessManagement → Core::Transition

ProcessManagement::Transition::Debug::Enabled

If enabled debugging information for transitions is logged.

This setting can not be deactivated.

Default value:

```
$Self->{'ProcessManagement::Transition::Debug::Enabled'} =  '0';
```

ProcessManagement::Transition::Debug::LogPriority

Defines the priority in which the information is logged and presented.

This setting is not active by default.

Default value:

```
$Self->{'ProcessManagement::Transition::Debug::LogPriority'} =  'debug';
```

ProcessManagement::Transition::Debug::Filter###00-Default

Filter for debugging Transitions. Note: More filters can be added in the format <OTRS_TICKET_Attribute> e.g. <OTRS_TICKET_Priority>.

This setting is not active by default.

Default value:

```
$Self->{'ProcessManagement::Transition::Debug::Filter'}->{'00-Default'} =  {
  '<OTRS_TICKET_TicketNumber>' => '',
  'TransitionEntityID' => ''
};
```

ProcessManagement → DynamicFields::Driver::Registration

DynamicFields::Driver###ProcessID

DynamicField backend registration.

Default value:

```
$Self->{'DynamicFields::Driver'}->{'ProcessID'} =  {
  'ConfigDialog' => 'AdminDynamicFieldText',
  'DisabledAdd' => '1',
  'DisplayName' => 'ProcessID',
  'Module' => 'Kernel::System::DynamicField::Driver::ProcessManagement::ProcessID'
};
```

DynamicFields::Driver###ActivityID
DynamicField backend registration.

Default value:

```
$Self->{'DynamicFields::Driver'}->{'ActivityID'} =  {
  'ConfigDialog' => 'AdminDynamicFieldText',
  'DisabledAdd' => '1',
  'DisplayName' => 'ActivityID',
  'Module' => 'Kernel::System::DynamicField::Driver::ProcessManagement::ActivityID'
};
```

ProcessManagement → Frontend::Admin::ModuleRegistration

Frontend::Module###AdminProcessManagement
Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminProcessManagement'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.ProcessManagement.css',
      'Core.AllocationList.css'
    ],
    'JavaScript' => [
      'thirdparty/jsplumb-1.6.4/jsplumb.js',
      'thirdparty/farahey-0.5/farahey.js',
      'thirdparty/jsplumb-labelspacer/label-spacer.js',
      'Core.Agent.Admin.ProcessManagement.js',
      'Core.Agent.Admin.ProcessManagement.Canvas.js',
      'Core.UI.AllocationList.js'
    ]
  },
  'NavBarModule' => {
    'Block' => 'System',
    'Description' => 'Configure Processes.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Process Management',
    'Prio' => '750'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Process Management'
};
```

Frontend::Module###AdminProcessManagementActivity
Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminProcessManagementActivity'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.ProcessManagement.css',
      'Core.AllocationList.css'
    ],
    'JavaScript' => [
      'Core.Agent.Admin.ProcessManagement.js',
      'Core.UI.AllocationList.js'
    ]
  },
  'Title' => 'Process Management Activity GUI'
```

```
};
```

Frontend::Module###AdminProcessManagementActivityDialog

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminProcessManagementActivityDialog'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.ProcessManagement.css',
      'Core.AllocationList.css'
    ],
    'JavaScript' => [
      'Core.Agent.Admin.ProcessManagement.js',
      'Core.UI.AllocationList.js'
    ]
  },
  'Title' => 'Process Management Activity Dialog GUI'
};
```

Frontend::Module###AdminProcessManagementTransition

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminProcessManagementTransition'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.ProcessManagement.css'
    ],
    'JavaScript' => [
      'Core.Agent.Admin.ProcessManagement.js'
    ]
  },
  'Title' => 'Process Management Transition GUI'
};
```

Frontend::Module###AdminProcessManagementTransitionAction

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminProcessManagementTransitionAction'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.ProcessManagement.css'
    ],
    'JavaScript' => [
      'Core.Agent.Admin.ProcessManagement.js'
    ]
  },
  'Title' => 'Process Management Transition Action GUI'
};
```

Frontend::Module###AdminProcessManagementPath

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminProcessManagementPath'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.ProcessManagement.css',
      'Core.AllocationList.css'
    ],
    'JavaScript' => [
      'Core.Agent.Admin.ProcessManagement.js',
      'Core.UI.AllocationList.js'
    ]
  },
  'Title' => 'Process Management Path GUI'
};
```

ProcessManagement → Frontend::Agent::Dashboard

DashboardBackend###0140-RunningTicketProcess

Parameters for the dashboard backend of the running process tickets overview of the agent interface . "Limit" is the number of entries shown by default. "Group" is used to restrict the access to the plugin (e. g. Group: admin;group1;group2;). "Default" determines if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTLLocal" is the cache time in minutes for the plugin.

Default value:

```
$Self->{'DashboardBackend'}->{'0140-RunningTicketProcess'} =  {
  'Attributes' => 'StateType=new;StateType=open;StateType=pending
reminder;StateType=pending auto',
  'Block' => 'ContentLarge',
  'CacheTTLLocal' => '0.5',
  'Default' => '0',
  'DefaultColumns' => {
    'Age' => '2',
    'Changed' => '1',
    'CustomerID' => '1',
    'CustomerName' => '1',
    'CustomerUserID' => '1',
    'DynamicField_ProcessManagementActivityID' => '2',
    'DynamicField_ProcessManagementProcessID' => '2',
    'EscalationResponseTime' => '1',
    'EscalationSolutionTime' => '1',
    'EscalationTime' => '1',
    'EscalationUpdateTime' => '1',
    'Lock' => '1',
    'Owner' => '1',
    'PendingTime' => '1',
    'Priority' => '1',
    'Queue' => '1',
    'Responsible' => '1',
    'SLA' => '1',
    'Service' => '1',
    'State' => '1',
    'TicketNumber' => '2',
    'Title' => '2',
    'Type' => '1'
  },
  'Description' => 'All tickets with a reminder set where the reminder date has been
reached',
  'Group' => '',
  'IsProcessWidget' => '1',
  'Limit' => '10',
  'Module' => 'Kernel::Output::HTML::Dashboard::TicketGeneric',
  'Permission' => 'rw',
  'Time' => 'UntilTime',
```

```
  'Title' => 'Running Process Tickets'
};
```

ProcessManagement → Frontend::Agent::ModuleRegistration

Frontend::Module###AgentTicketProcess

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketProcess'} =  {
  'Description' => 'Create new process ticket.',
  'Loader' => {
    'CSS' => [
      'Core.Agent.TicketProcess.css'
    ],
    'JavaScript' => [
      'Core.Agent.CustomerSearch.js',
      'Core.Agent.TicketAction.js',
      'Core.Agent.TicketProcess.js'
    ]
  },
  'NavBar' => [
    {
      'AccessKey' => 'p',
      'Block' => '',
      'Description' => 'Create New process ticket.',
      'Link' => 'Action=AgentTicketProcess',
      'LinkOption' => '',
      'Name' => 'New process ticket',
      'NavBar' => 'Ticket',
      'Prio' => '220',
      'Type' => ''
    }
  ],
  'NavBarName' => 'Ticket',
  'Title' => 'New process ticket'
};
```

ProcessManagement → Frontend::Agent::NavBarModule

Frontend::NavBarModule###1-TicketProcesses

Frontend module registration (disable ticket processes screen if no process available).

Default value:

```
$Self->{'Frontend::NavBarModule'}->{'1-TicketProcesses'} =  {
  'Module' => 'Kernel::Output::HTML::NavBar::AgentTicketProcess'
};
```

ProcessManagement → Frontend::Agent::Ticket::MenuModule

Ticket::Frontend::MenuModule###480-Process

Shows a link in the menu to enroll a ticket into a process in the ticket zoom view of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::MenuModule'}->{'480-Process'} =  {
  'Action' => 'AgentTicketProcess',
  'Cluster' => '',
  'Description' => 'Enroll process for this ticket',
  'Link' => 'Action=AgentTicketProcess;IsProcessEnroll=1;TicketID=[% Data.TicketID |
html %]',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Process',
  'Name' => 'Process',
  'PopupType' => 'TicketAction',
  'Target' => ''
```

```
};
```

ProcessManagement → Frontend::Agent::Ticket::ViewProcess

Ticket::Frontend::AgentTicketProcess###StateType
Determines the next possible ticket states, for process tickets in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketProcess'}->{'StateType'} =  [
  'new',
  'open',
  'pending auto',
  'pending reminder',
  'closed'
];
```

Ticket::Frontend::CustomerTicketProcess###StateType
Determines the next possible ticket states, for process tickets in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketProcess'}->{'StateType'} =  [
  'new',
  'open'
];
```

Ticket::Frontend::AgentTicketProcess::CustomerIDReadOnly
Controls if CutomerID is editable in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketProcess::CustomerIDReadOnly'} =  '1';
```

ProcessManagement → Frontend::Agent::Ticket::ViewZoom

Ticket::Frontend::AgentTicketZoom###ProcessDisplay
Display settings to override defaults for Process Tickets.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketZoom'}->{'ProcessDisplay'} =  {
  'NavBarName' => 'Processes',
  'WidgetTitle' => 'Process Information'
};
```

Ticket::Frontend::AgentTicketZoom###ProcessWidgetDynamicFieldGroups
Dynamic fields groups for process widget. The key is the name of the group, the value contains the fields to be shown. Example: 'Key => My Group', 'Content: Name_X, NameY'.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketZoom'}->{'ProcessWidgetDynamicFieldGroups'} =
{};
```

Ticket::Frontend::AgentTicketZoom###ProcessWidgetDynamicField
Dynamic fields shown in the process widget in ticket zoom screen of the agent interface. Possible settings: 0 = Disabled, 1 = Enabled.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketZoom'}->{'ProcessWidgetDynamicField'} =  {};
```

ProcessManagement → Frontend::Customer::ModuleRegistration

CustomerFrontend::Module###CustomerTicketProcess
Frontend module registration for the customer interface.

Default value:

```
$Self->{'CustomerFrontend::Module'}->{'CustomerTicketProcess'} =  {
  'Description' => 'Process Ticket.',
  'Loader' => {
    'CSS' => [
      'Core.Customer.TicketProcess.css'
    ]
  },
  'NavBar' => [
    {
      'AccessKey' => 'o',
      'Block' => '',
      'Description' => 'Create new process ticket.',
      'Link' => 'Action=CustomerTicketProcess',
      'LinkOption' => '',
      'Name' => 'New process ticket',
      'NavBar' => 'Ticket',
      'Prio' => '220',
      'Type' => 'Submenu'
    }
  ],
  'NavBarName' => 'Ticket',
  'Title' => 'Process ticket'
};
```

ProcessManagement → Frontend::Customer::NavBarModule

CustomerFrontend::NavBarModule###10-CustomerTicketProcesses
Frontend module registration (disable ticket processes screen if no process available) for Customer.

Default value:

```
$Self->{'CustomerFrontend::NavBarModule'}->{'10-CustomerTicketProcesses'} =  {
  'Module' => 'Kernel::Output::HTML::NavBar::CustomerTicketProcess'
};
```

7. Ticket

Ticket → Core::CustomerCompany

CustomerCompany::EventModulePost###110-UpdateTickets
Event module that updates tickets after an update of the Customer.

Default value:

```
$Self->{'CustomerCompany::EventModulePost'}->{'110-UpdateTickets'} =  {
  'Event' => 'CustomerCompanyUpdate',
  'Module' => 'Kernel::System::CustomerCompany::Event::TicketUpdate',
  'Transaction' => '0'
};
```

CustomerUser::EventModulePost###120-UpdateTickets

Event module that updates tickets after an update of the Customer User.

Default value:

```
$Self->{'CustomerUser::EventModulePost'}->{'120-UpdateTickets'} =  {
  'Event' => 'CustomerUserUpdate',
  'Module' => 'Kernel::System::CustomerUser::Event::TicketUpdate',
  'Transaction' => '0'
};
```

Ticket → Core::FulltextSearch

Ticket::SearchIndexModule

Helps to extend your articles full-text search (From, To, Cc, Subject and Body search). Runtime will do full-text searches on live data (it works fine for up to 50.000 tickets). StaticDB will strip all articles and will build an index after article creation, increasing fulltext searches about 50%. To create an initial index use "bin/otrs.Console.pl Maint::Ticket::FulltextIndexRebuild".

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::SearchIndexModule'} =
'Kernel::System::Ticket::ArticleSearchIndex::RuntimeDB';
```

Ticket::SearchIndex::WarnOnStopWordUsage

Display a warning and prevent search when using stop words within fulltext search.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::SearchIndex::WarnOnStopWordUsage'} =  '0';
```

Ticket::SearchIndex::Attribute

Basic fulltext index settings. Execute "bin/otrs.Console.pl Maint::Ticket::FulltextIndexRebuild" in order to generate a new index.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::SearchIndex::Attribute'} =  {
  'WordCountMax' => '1000',
  'WordLengthMax' => '30',
  'WordLengthMin' => '3'
};
```

Ticket::SearchIndex::Filters

Fulltext index regex filters to remove parts of the text.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::SearchIndex::Filters'} =  [
  '[,\\&\\<\\>\\?"\\!\\*\\|;\\[\\]\\(\\)\\+\\$\\^=]',
  '^[\':.]|[\':.]$',
  '^[^\\w]+$'
];
```

Ticket::SearchIndex::StopWords###en

English stop words for fulltext index. These words will be removed from the search index.

Default value:

```
$Self->{'Ticket::SearchIndex::StopWords'}->{'en'} =  [
 'a',
 'about',
 'above',
 'after',
 'again',
 'against',
 'all',
 'am',
 'an',
 'and',
 'any',
 'are',
 'aren\'t',
 'as',
 'at',
 'be',
 'because',
 'been',
 'before',
 'being',
 'below',
 'between',
 'both',
 'but',
 'by',
 'can\'t',
 'cannot',
 'could',
 'couldn\'t',
 'did',
 'didn\'t',
 'do',
 'does',
 'doesn\'t',
 'doing',
 'don\'t',
 'down',
 'during',
 'each',
 'few',
 'for',
 'from',
 'further',
 'had',
 'hadn\'t',
 'has',
 'hasn\'t',
 'have',
 'haven\'t',
 'having',
 'he',
 'he\'d',
 'he\'ll',
 'he\'s',
 'her',
 'here',
 'here\'s',
 'hers',
 'herself',
 'him',
 'himself',
 'his',
 'how',
 'how\'s',
 'i',
 'i\'d',
 'i\'ll',
 'i\'m',
 'i\'ve',
```

```
'if',
'in',
'into',
'is',
'isn\'t',
'it',
'it\'s',
'its',
'itself',
'let\'s',
'me',
'more',
'most',
'mustn\'t',
'my',
'myself',
'no',
'nor',
'not',
'of',
'off',
'on',
'once',
'only',
'or',
'other',
'ought',
'our',
'ours',
'ourselves',
'out',
'over',
'own',
'same',
'shan\'t',
'she',
'she\'d',
'she\'ll',
'she\'s',
'should',
'shouldn\'t',
'so',
'some',
'such',
'than',
'that',
'that\'s',
'the',
'their',
'theirs',
'them',
'themselves',
'then',
'there',
'there\'s',
'these',
'they',
'they\'d',
'they\'ll',
'they\'re',
'they\'ve',
'this',
'those',
'through',
'to',
'too',
'under',
'until',
'up',
'very',
'was',
'wasn\'t',
```

```
  'we',
  'we\'d',
  'we\'ll',
  'we\'re',
  'we\'ve',
  'were',
  'weren\'t',
  'what',
  'what\'s',
  'when',
  'when\'s',
  'where',
  'where\'s',
  'which',
  'while',
  'who',
  'who\'s',
  'whom',
  'why',
  'why\'s',
  'with',
  'won\'t',
  'would',
  'wouldn\'t',
  'you',
  'you\'d',
  'you\'ll',
  'you\'re',
  'you\'ve',
  'your',
  'yours',
  'yourself',
  'yourselves'
];
```

Ticket::SearchIndex::StopWords###de

German stop words for fulltext index. These words will be removed from the search index.

Default value:

```
$Self->{'Ticket::SearchIndex::StopWords'}->{'de'} =  [
  'aber',
  'als',
  'am',
  'an',
  'auch',
  'auf',
  'aus',
  'bei',
  'bin',
  'bis',
  'bist',
  'da',
  'dadurch',
  'daher',
  'darum',
  'das',
  'daß',
  'dass',
  'dein',
  'deine',
  'dem',
  'den',
  'der',
  'des',
  'dessen',
  'deshalb',
  'die',
  'dies',
  'dieser',
  'dieses',
```

```
'doch',
'dort',
'du',
'durch',
'ein',
'eine',
'einem',
'einen',
'einer',
'eines',
'er',
'es',
'euer',
'eure',
'für',
'hatte',
'hatten',
'hattest',
'hattet',
'hier',
'hinter',
'ich',
'ihr',
'ihre',
'im',
'in',
'ist',
'ja',
'jede',
'jedem',
'jeden',
'jeder',
'jedes',
'jener',
'jenes',
'jetzt',
'kann',
'kannst',
'können',
'könnt',
'machen',
'mein',
'meine',
'mit',
'muß',
'mußt',
'musst',
'müssen',
'müßt',
'nach',
'nachdem',
'nein',
'nicht',
'nun',
'oder',
'seid',
'sein',
'seine',
'sich',
'sie',
'sind',
'soll',
'sollen',
'sollst',
'sollt',
'sonst',
'soweit',
'sowie',
'und',
'unser',
'unsere',
'unter',
```

```
  'vom',
  'von',
  'vor',
  'wann',
  'warum',
  'was',
  'weiter',
  'weitere',
  'wenn',
  'wer',
  'werde',
  'werden',
  'werdet',
  'weshalb',
  'wie',
  'wieder',
  'wieso',
  'wir',
  'wird',
  'wirst',
  'wo',
  'woher',
  'wohin',
  'zu',
  'zum',
  'zur',
  'über'
];
```

Ticket::SearchIndex::StopWords###nl

Dutch stop words for fulltext index. These words will be removed from the search index.

Default value:

```
$Self->{'Ticket::SearchIndex::StopWords'}->{'nl'} =  [
  'de',
  'zijn',
  'een',
  'en',
  'in',
  'je',
  'het',
  'van',
  'op',
  'ze',
  'hebben',
  'het',
  'hij',
  'niet',
  'met',
  'er',
  'dat',
  'die',
  'te',
  'wat',
  'voor',
  'naar',
  'gaan',
  'kunnen',
  'zeggen',
  'dat',
  'maar',
  'aan',
  'veel',
  'zijn',
  'worden',
  'uit',
  'ook',
  'komen',
  'als',
  'om',
```

```
'moeten',
'we',
'doen',
'bij',
'goed',
'haar',
'dan',
'nog',
'of',
'maken',
'zo',
'wel',
'mijn',
'zien',
'over',
'willen',
'staan',
'door',
'kijken',
'zullen',
'heel',
'nu',
'weten',
'zitten',
'hem',
'schrijven',
'vinden',
'woord',
'hoe',
'geen',
'dit',
'mens',
'al',
'jij',
'ander',
'groot',
'waar',
'maar',
'weer',
'kind',
'me',
'vragen',
'een',
'denken',
'twee',
'horen',
'iets',
'deze',
'krijgen',
'ons',
'zich',
'lezen',
'hun',
'welk',
'zin',
'laten',
'mogen',
'hier',
'jullie',
'toch',
'geven',
'jaar',
'tegen',
'al',
'eens',
'echt',
'houden',
'alleen',
'lopen',
'mee',
'ja',
'roepen',
```

```
  'tijd',
  'dag',
  'elkaar',
  'even',
  'lang',
  'land',
  'liggen',
  'waarom',
  'zetten',
  'vader',
  'laat',
  'beginnen',
  'blijven',
  'nee',
  'moeder',
  'huis',
  'nou',
  'na',
  'af',
  'keer',
  'dus',
  'tot',
  'vertellen',
  'wie',
  'net',
  'jou',
  'les',
  'want',
  'man',
  'nieuw',
  'elk',
  'tekst',
  'omdat',
  'gebruiken',
  'u'
];
```

Ticket::SearchIndex::StopWords###es

Spanish stop words for fulltext index. These words will be removed from the search index.

Default value:

```
$Self->{'Ticket::SearchIndex::StopWords'}->{'es'} =  [
  'un',
  'una',
  'unas',
  'unos',
  'uno',
  'sobre',
  'todo',
  'también',
  'tras',
  'otro',
  'algún',
  'alguno',
  'alguna',
  'algunos',
  'algunas',
  'ser',
  'es',
  'soy',
  'eres',
  'somos',
  'sois',
  'estoy',
  'esta',
  'estamos',
  'estais',
  'estan',
  'como',
  'en',
```

```
'para',
'atras',
'porque',
'por qué',
'estado',
'estaba',
'ante',
'antes',
'siendo',
'ambos',
'pero',
'por',
'poder',
'puede',
'puedo',
'podemos',
'podeis',
'pueden',
'fui',
'fue',
'fuimos',
'fueron',
'hacer',
'hago',
'hace',
'hacemos',
'haceis',
'hacen',
'cada',
'fin',
'incluso',
'primero',
'desde',
'conseguir',
'consigo',
'consigue',
'consigues',
'conseguimos',
'consiguen',
'ir',
'voy',
'va',
'vamos',
'vais',
'van',
'vaya',
'gueno',
'ha',
'tener',
'tengo',
'tiene',
'tenemos',
'teneis',
'tienen',
'el',
'la',
'lo',
'las',
'los',
'su',
'aqui',
'mio',
'tuyo',
'ellos',
'ellas',
'nos',
'nosotros',
'vosotros',
'vosotras',
'si',
'dentro',
'solo',
```

```
'solamente',
'saber',
'sabes',
'sabe',
'sabemos',
'sabeis',
'saben',
'ultimo',
'largo',
'bastante',
'haces',
'muchos',
'aquellos',
'aquellas',
'sus',
'entonces',
'tiempo',
'verdad',
'verdadero',
'verdadera',
'cierto',
'ciertos',
'cierta',
'ciertas',
'intentar',
'intento',
'intenta',
'intentas',
'intentamos',
'intentais',
'intentan',
'dos',
'bajo',
'arriba',
'encima',
'usar',
'uso',
'usas',
'usa',
'usamos',
'usais',
'usan',
'emplear',
'empleo',
'empleas',
'emplean',
'ampleamos',
'empleais',
'valor',
'muy',
'era',
'eras',
'eramos',
'eran',
'modo',
'bien',
'cual',
'cuando',
'donde',
'mientras',
'quien',
'con',
'entre',
'sin',
'trabajo',
'trabajar',
'trabajas',
'trabaja',
'trabajamos',
'trabajais',
'trabajan',
'podria',
```

```
  'podrias',
  'podriamos',
  'podrian',
  'podriais',
  'yo',
  'aquel'
];
```

Ticket::SearchIndex::StopWords###fr

French stop words for fulltext index. These words will be removed from the search index.

Default value:

```
$Self->{'Ticket::SearchIndex::StopWords'}->{'fr'} =  [
  'alors',
  'au',
  'aucuns',
  'aussi',
  'autre',
  'avant',
  'avec',
  'avoir',
  'bon',
  'car',
  'ce',
  'cela',
  'ces',
  'ceux',
  'chaque',
  'ci',
  'comme',
  'comment',
  'dans',
  'des',
  'du',
  'dedans',
  'dehors',
  'depuis',
  'deux',
  'devrait',
  'doit',
  'donc',
  'dos',
  'droite',
  'début',
  'elle',
  'elles',
  'en',
  'encore',
  'essai',
  'est',
  'et',
  'eu',
  'fait',
  'faites',
  'fois',
  'font',
  'force',
  'haut',
  'hors',
  'ici',
  'il',
  'ils',
  'je',
  'juste',
  'la',
  'le',
  'les',
  'leur',
  'là',
  'ma',
```

```
  'maintenant',
  'mais',
  'mes',
  'mine',
  'moins',
  'mon',
  'mot',
  'même',
  'ni',
  'nommés',
  'notre',
  'nous',
  'nouveaux',
  'ou',
  'où',
  'par',
  'parce',
  'parole',
  'pas',
  'personnes',
  'peut',
  'peu',
  'pièce',
  'plupart',
  'pour',
  'pourquoi',
  'quand',
  'que',
  'quel',
  'quelle',
  'quelles',
  'quels',
  'qui',
  'sa',
  'sans',
  'ses',
  'seulement',
  'si',
  'sien',
  'son',
  'sont',
  'sous',
  'soyez',
  'sujet',
  'sur',
  'ta',
  'tandis',
  'tellement',
  'tels',
  'tes',
  'ton',
  'tous',
  'tout',
  'trop',
  'très',
  'tu',
  'valeur',
  'voie',
  'voient',
  'vont',
  'votre',
  'vous',
  'vu',
  'ça',
  'étaient',
  'état',
  'étions',
  'été',
  'être'
];
```

Ticket::SearchIndex::StopWords###it

Italian stop words for fulltext index. These words will be removed from the search index.

Default value:

```
$Self->{'Ticket::SearchIndex::StopWords'}->{'it'} =  [
 'a',
 'adesso',
 'ai',
 'al',
 'alla',
 'allo',
 'allora',
 'altre',
 'altri',
 'altro',
 'anche',
 'ancora',
 'avere',
 'aveva',
 'avevano',
 'ben',
 'buono',
 'che',
 'chi',
 'cinque',
 'comprare',
 'con',
 'consecutivi',
 'consecutivo',
 'cosa',
 'cui',
 'da',
 'del',
 'della',
 'dello',
 'dentro',
 'deve',
 'devo',
 'di',
 'doppio',
 'due',
 'e',
 'ecco',
 'fare',
 'fine',
 'fino',
 'fra',
 'gente',
 'giu',
 'ha',
 'hai',
 'hanno',
 'ho',
 'il',
 'indietro',
 'invece',
 'io',
 'la',
 'lavoro',
 'le',
 'lei',
 'lo',
 'loro',
 'lui',
 'lungo',
 'ma',
 'me',
 'meglio',
 'molta',
 'molti',
```

```
    'molto',
    'nei',
    'nella',
    'no',
    'noi',
    'nome',
    'nostro',
    'nove',
    'nuovi',
    'nuovo',
    'o',
    'oltre',
    'ora',
    'otto',
    'peggio',
    'pero',
    'persone',
    'piu',
    'poco',
    'primo',
    'promesso',
    'qua',
    'quarto',
    'quasi',
    'quattro',
    'quello',
    'questo',
    'qui',
    'quindi',
    'quinto',
    'rispetto',
    'sara',
    'secondo',
    'sei',
    'sembra',
    'sembrava',
    'senza',
    'sette',
    'sia',
    'siamo',
    'siete',
    'solo',
    'sono',
    'sopra',
    'soprattutto',
    'sotto',
    'stati',
    'stato',
    'stesso',
    'su',
    'subito',
    'sul',
    'sulla',
    'tanto',
    'te',
    'tempo',
    'terzo',
    'tra',
    'tre',
    'triplo',
    'ultimo',
    'un',
    'una',
    'uno',
    'va',
    'vai',
    'voi',
    'volte',
    'vostro'
];
```

Ticket::SearchIndex::StopWords###Custom
Customizable stop words for fulltext index. These words will be removed from the search index.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::SearchIndex::StopWords'}->{'Custom'} =  [
  'MyStopWord'
];
```

Ticket::EventModulePost###098-ArticleSearchIndex
Builds an article index right after the article's creation.

Default value:

```
$Self->{'Ticket::EventModulePost'}->{'098-ArticleSearchIndex'} =  {
  'Event' => '(ArticleCreate|ArticleUpdate)',
  'Module' => 'Kernel::System::Ticket::Event::ArticleSearchIndex'
};
```

Ticket → Core::LinkObject

LinkObject::PossibleLink###0200
Links 2 tickets with a "Normal" type link.

Default value:

```
$Self->{'LinkObject::PossibleLink'}->{'0200'} =  {
  'Object1' => 'Ticket',
  'Object2' => 'Ticket',
  'Type' => 'Normal'
};
```

LinkObject::PossibleLink###0201
Links 2 tickets with a "ParentChild" type link.

Default value:

```
$Self->{'LinkObject::PossibleLink'}->{'0201'} =  {
  'Object1' => 'Ticket',
  'Object2' => 'Ticket',
  'Type' => 'ParentChild'
};
```

LinkObject::IgnoreLinkedTicketStateTypes
Defines, which tickets of which ticket state types should not be listed in linked ticket lists.

Default value:

```
$Self->{'LinkObject::IgnoreLinkedTicketStateTypes'} =  [
  'merged',
  'removed'
];
```

Ticket → Core::PostMaster

PostmasterMaxEmails
Maximal auto email responses to own email-address a day (Loop-Protection).

This setting can not be deactivated.

Default value:

```
$Self->{'PostmasterMaxEmails'} =  '40';
```

PostMasterMaxEmailSize
Maximal size in KBytes for mails that can be fetched via POP3/POP3S/IMAP/IMAPS (KBytes).

This setting can not be deactivated.

Default value:

```
$Self->{'PostMasterMaxEmailSize'} =  '16384';
```

PostMasterReconnectMessage
The maximum number of mails fetched at once before reconnecting to the server.

This setting can not be deactivated.

Default value:

```
$Self->{'PostMasterReconnectMessage'} =  '20';
```

LoopProtectionModule
Default loop protection module.

This setting can not be deactivated.

Default value:

```
$Self->{'LoopProtectionModule'} =  'Kernel::System::PostMaster::LoopProtection::DB';
```

LoopProtectionLog
Path for the log file (it only applies if "FS" was selected for LoopProtectionModule and it is mandatory).

This setting can not be deactivated.

Default value:

```
$Self->{'LoopProtectionLog'} =  '<OTRS_CONFIG_Home>/var/log/LoopProtection';
```

PostmasterAutoHTML2Text
Converts HTML mails into text messages.

This setting can not be deactivated.

Default value:

```
$Self->{'PostmasterAutoHTML2Text'} =  '1';
```

PostmasterUserID
Specifies user id of the postmaster data base.

This setting can not be deactivated.

Default value:

```
$Self->{'PostmasterUserID'} =  '1';
```

PostmasterDefaultQueue
Defines the postmaster default queue.

This setting can not be deactivated.

Default value:

```
$Self->{'PostmasterDefaultQueue'} =  'Raw';
```

PostmasterDefaultPriority
Defines the default priority of new tickets.

This setting can not be deactivated.

Default value:

```
$Self->{'PostmasterDefaultPriority'} =  '3 normal';
```

PostmasterDefaultState
Defines the default state of new tickets.

This setting can not be deactivated.

Default value:

```
$Self->{'PostmasterDefaultState'} =  'new';
```

PostmasterFollowUpState
Defines the state of a ticket if it gets a follow-up.

This setting can not be deactivated.

Default value:

```
$Self->{'PostmasterFollowUpState'} =  'open';
```

PostmasterFollowUpStateClosed
Defines the state of a ticket if it gets a follow-up and the ticket was already closed.

This setting is not active by default.

Default value:

```
$Self->{'PostmasterFollowUpStateClosed'} =  'open';
```

PostmasterFollowUpOnUnlockAgentNotifyOnlyToOwner
Sends agent follow-up notification only to the owner, if a ticket is unlocked (the default is to send the notification to all agents).

This setting can not be deactivated.

Default value:

```
$Self->{'PostmasterFollowUpOnUnlockAgentNotifyOnlyToOwner'} =  '0';
```

PostmasterHeaderFieldCount
Defines the number of header fields in frontend modules for add and update postmaster filters. It can be up to 99 fields.

This setting can not be deactivated.

Default value:

```
$Self->{'PostmasterHeaderFieldCount'} =  '12';
```

PostmasterX-Header
Defines all the X-headers that should be scanned.

This setting can not be deactivated.

Default value:

```
$Self->{'PostmasterX-Header'} =  [
 'From',
 'To',
 'Cc',
 'Reply-To',
 'ReplyTo',
 'Subject',
```

```
  'Message-ID',
  'Message-Id',
  'Resent-To',
  'Resent-From',
  'Precedence',
  'Mailing-List',
  'List-Id',
  'List-Archive',
  'Errors-To',
  'References',
  'In-Reply-To',
  'Auto-Submitted',
  'X-Loop',
  'X-Spam-Flag',
  'X-Spam-Level',
  'X-Spam-Score',
  'X-Spam-Status',
  'X-No-Loop',
  'X-Priority',
  'Importance',
  'X-Mailer',
  'User-Agent',
  'Organization',
  'X-Original-To',
  'Delivered-To',
  'Envelope-To',
  'X-Envelope-To',
  'Return-Path',
  'X-OTRS-Owner',
  'X-OTRS-OwnerID',
  'X-OTRS-Responsible',
  'X-OTRS-ResponsibleID',
  'X-OTRS-Loop',
  'X-OTRS-Priority',
  'X-OTRS-Queue',
  'X-OTRS-Lock',
  'X-OTRS-Ignore',
  'X-OTRS-State',
  'X-OTRS-State-PendingTime',
  'X-OTRS-Type',
  'X-OTRS-Service',
  'X-OTRS-SLA',
  'X-OTRS-CustomerNo',
  'X-OTRS-CustomerUser',
  'X-OTRS-SenderType',
  'X-OTRS-ArticleType',
  'X-OTRS-FollowUp-Priority',
  'X-OTRS-FollowUp-Queue',
  'X-OTRS-FollowUp-Lock',
  'X-OTRS-FollowUp-State',
  'X-OTRS-FollowUp-State-PendingTime',
  'X-OTRS-FollowUp-Type',
  'X-OTRS-FollowUp-Service',
  'X-OTRS-FollowUp-SLA',
  'X-OTRS-FollowUp-SenderType',
  'X-OTRS-FollowUp-ArticleType',
  'X-OTRS-BodyDecrypted'
];
```

PostMaster::PreFilterModule###1-Match

Module to filter and manipulate incoming messages. Block/ignore all spam email with From: noreply@ address.

This setting is not active by default.

Default value:

```
$Self->{'PostMaster::PreFilterModule'}->{'1-Match'} =  {
  'Match' => {
    'From' => 'noreply@'
  },
```

```
  'Module' => 'Kernel::System::PostMaster::Filter::Match',
  'Set' => {
    'X-OTRS-Ignore' => 'yes'
  }
};
```

PostMaster::PreFilterModule###2-Match

Module to filter and manipulate incoming messages. Get a 4 digit number to ticket free text, use regex in Match e. g. From => '(.+?)@.+?', and use () as [***] in Set =>.

This setting is not active by default.

Default value:

```
$Self->{'PostMaster::PreFilterModule'}->{'2-Match'} =  {
  'Match' => {
    'Subject' => 'SomeNumber:(\\d\\d\\d\\d)'
  },
  'Module' => 'Kernel::System::PostMaster::Filter::Match',
  'Set' => {
    'X-OTRS-DynamicField-TicketFreeKey1' => 'SomeNumber',
    'X-OTRS-DynamicField-TicketFreeText1' => '[***]'
  }
};
```

PostMaster::PreFilterModule###3-NewTicketReject

Blocks all the incoming emails that do not have a valid ticket number in subject with From: @example.com address.

This setting is not active by default.

Default value:

```
$Self->{'PostMaster::PreFilterModule'}->{'3-NewTicketReject'} =  {
  'Match' => {
    'From' => '@example.com'
  },
  'Module' => 'Kernel::System::PostMaster::Filter::NewTicketReject',
  'Set' => {
    'X-OTRS-Ignore' => 'yes'
  }
};
```

PostMaster::PreFilterModule::NewTicketReject::Sender

Defines the sender for rejected emails.

This setting is not active by default.

Default value:

```
$Self->{'PostMaster::PreFilterModule::NewTicketReject::Sender'} =
'noreply@example.com';
```

PostMaster::PreFilterModule::NewTicketReject::Subject

Defines the subject for rejected emails.

This setting can not be deactivated.

Default value:

```
$Self->{'PostMaster::PreFilterModule::NewTicketReject::Subject'} =  'Email Rejected';
```

PostMaster::PreFilterModule::NewTicketReject::Body

Defines the body text for rejected emails.

This setting can not be deactivated.

Default value:

```
 $Self->{'PostMaster::PreFilterModule::NewTicketReject::Body'} =  '
Dear Customer,

Unfortunately we could not detect a valid ticket number
in your subject, so this email can\'t be processed.

Please create a new ticket via the customer panel.

Thanks for your help!

 Your Helpdesk Team
';
```

PostMaster::PreFilterModule###4-CMD

CMD example setup. Ignores emails where external CMD returns some output on STD-OUT (email will be piped into STDIN of some.bin).

This setting is not active by default.

Default value:

```
 $Self->{'PostMaster::PreFilterModule'}->{'4-CMD'} =  {
  'CMD' => '/usr/bin/some.bin',
  'Module' => 'Kernel::System::PostMaster::Filter::CMD',
  'Set' => {
    'X-OTRS-Ignore' => 'yes'
  }
};
```

PostMaster::PreFilterModule###5-SpamAssassin

Spam Assassin example setup. Ignores emails that are marked with SpamAssassin.

This setting is not active by default.

Default value:

```
 $Self->{'PostMaster::PreFilterModule'}->{'5-SpamAssassin'} =  {
  'CMD' => '/usr/bin/spamassassin | grep -i "X-Spam-Status: yes"',
  'Module' => 'Kernel::System::PostMaster::Filter::CMD',
  'Set' => {
    'X-OTRS-Ignore' => 'yes'
  }
};
```

PostMaster::PreFilterModule###6-SpamAssassin

Spam Assassin example setup. Moves marked mails to spam queue.

This setting is not active by default.

Default value:

```
 $Self->{'PostMaster::PreFilterModule'}->{'6-SpamAssassin'} =  {
  'CMD' => '/usr/bin/spamassassin | grep -i "X-Spam-Status: yes"',
  'Module' => 'Kernel::System::PostMaster::Filter::CMD',
  'Set' => {
    'X-OTRS-Queue' => 'spam'
  }
};
```

PostMaster::PreFilterModule###000-MatchDBSource

Module to use database filter storage.

This setting can not be deactivated.

Default value:

```
 $Self->{'PostMaster::PreFilterModule'}->{'000-MatchDBSource'} =  {
  'Module' => 'Kernel::System::PostMaster::Filter::MatchDBSource'
};
```

PostMaster::PostFilterModule###000-FollowUpArticleTypeCheck

Module to check if arrived emails should be marked as email-internal (because of original forwarded internal email). ArticleType and SenderType define the values for the arrived email/article.

Default value:

```
$Self->{'PostMaster::PostFilterModule'}->{'000-FollowUpArticleTypeCheck'} =  {
  'ArticleType' => 'email-internal',
  'Module' => 'Kernel::System::PostMaster::Filter::FollowUpArticleTypeCheck',
  'SenderType' => 'customer'
};
```

PostMaster::PreFilterModule###000-ExternalTicketNumberRecognition1

Recognize if a ticket is a follow-up to an existing ticket using an external ticket number.

This setting is not active by default.

Default value:

```
$Self->{'PostMaster::PreFilterModule'}->{'000-ExternalTicketNumberRecognition1'} =  {
  'ArticleType' => 'note-report',
  'DynamicFieldName' => 'Name_X',
  'FromAddressRegExp' => '\\s*@example.com',
  'Module' => 'Kernel::System::PostMaster::Filter::ExternalTicketNumberRecognition',
  'Name' => 'Some Description',
  'NumberRegExp' => '\\s*Incident-(\\d.*)\\s*',
  'SearchInBody' => '1',
  'SearchInSubject' => '1',
  'SenderType' => 'system',
  'TicketStateTypes' => 'new;open'
};
```

PostMaster::PreFilterModule###000-ExternalTicketNumberRecognition2

Recognize if a ticket is a follow-up to an existing ticket using an external ticket number.

This setting is not active by default.

Default value:

```
$Self->{'PostMaster::PreFilterModule'}->{'000-ExternalTicketNumberRecognition2'} =  {
  'ArticleType' => 'note-report',
  'DynamicFieldName' => 'Name_X',
  'FromAddressRegExp' => '\\s*@example.com',
  'Module' => 'Kernel::System::PostMaster::Filter::ExternalTicketNumberRecognition',
  'Name' => 'Some Description',
  'NumberRegExp' => '\\s*Incident-(\\d.*)\\s*',
  'SearchInBody' => '1',
  'SearchInSubject' => '1',
  'SenderType' => 'system',
  'TicketStateTypes' => 'new;open'
};
```

PostMaster::PreFilterModule###000-ExternalTicketNumberRecognition3

Recognize if a ticket is a follow-up to an existing ticket using an external ticket number.

This setting is not active by default.

Default value:

```
$Self->{'PostMaster::PreFilterModule'}->{'000-ExternalTicketNumberRecognition3'} =  {
  'ArticleType' => 'note-report',
  'DynamicFieldName' => 'Name_X',
  'FromAddressRegExp' => '\\s*@example.com',
  'Module' => 'Kernel::System::PostMaster::Filter::ExternalTicketNumberRecognition',
  'Name' => 'Some Description',
  'NumberRegExp' => '\\s*Incident-(\\d.*)\\s*',
  'SearchInBody' => '1',
  'SearchInSubject' => '1',
```

```
  'SenderType' => 'system',
  'TicketStateTypes' => 'new;open'
};
```

PostMaster::PreFilterModule###000-ExternalTicketNumberRecognition4
Recognize if a ticket is a follow-up to an existing ticket using an external ticket number.

This setting is not active by default.

Default value:

```
$Self->{'PostMaster::PreFilterModule'}->{'000-ExternalTicketNumberRecognition4'} =  {
  'ArticleType' => 'note-report',
  'DynamicFieldName' => 'Name_X',
  'FromAddressRegExp' => '\\s*@example.com',
  'Module' => 'Kernel::System::PostMaster::Filter::ExternalTicketNumberRecognition',
  'Name' => 'Some Description',
  'NumberRegExp' => '\\s*Incident-(\\d.*)\\s*',
  'SearchInBody' => '1',
  'SearchInSubject' => '1',
  'SenderType' => 'system',
  'TicketStateTypes' => 'new;open'
};
```

PostMaster::PreFilterModule###000-DecryptBody
Module to filter encrypted bodies of incoming messages.

This setting is not active by default.

Default value:

```
$Self->{'PostMaster::PreFilterModule'}->{'000-DecryptBody'} =  {
  'Module' => 'Kernel::System::PostMaster::Filter::Decrypt',
  'StoreDecryptedBody' => '0'
};
```

PostMaster::PreFilterModule###000-SMIMEFetchFromCustomer
Module to fetch customer users SMIME certificates of incoming messages.

Default value:

```
$Self->{'PostMaster::PreFilterModule'}->{'000-SMIMEFetchFromCustomer'} =  {
  'Module' => 'Kernel::System::PostMaster::Filter::SMIMEFetchFromCustomer'
};
```

PostMaster::CheckFollowUpModule###0100-Subject
Checks if an E-Mail is a followup to an existing ticket by searching the subject for a valid ticket number.

Default value:

```
$Self->{'PostMaster::CheckFollowUpModule'}->{'0100-Subject'} =  {
  'Module' => 'Kernel::System::PostMaster::FollowUpCheck::Subject'
};
```

PostMaster::CheckFollowUpModule###0200-References
Executes follow-up checks on In-Reply-To or References headers for mails that don't have a ticket number in the subject.

Default value:

```
$Self->{'PostMaster::CheckFollowUpModule'}->{'0200-References'} =  {
  'Module' => 'Kernel::System::PostMaster::FollowUpCheck::References'
};
```

PostMaster::CheckFollowUpModule###0300-Body
Executes follow-up checks on email body for mails that don't have a ticket number in the subject.

This setting is not active by default.

Default value:

```
$Self->{'PostMaster::CheckFollowUpModule'}->{'0300-Body'} =  {
  'Module' => 'Kernel::System::PostMaster::FollowUpCheck::Body'
};
```

PostMaster::CheckFollowUpModule###0400-Attachments

Executes follow-up checks on attachment contents for mails that don't have a ticket number in the subject.

This setting is not active by default.

Default value:

```
$Self->{'PostMaster::CheckFollowUpModule'}->{'0400-Attachments'} =  {
  'Module' => 'Kernel::System::PostMaster::FollowUpCheck::Attachments'
};
```

PostMaster::CheckFollowUpModule###0500-RawEmail

Executes follow-up checks on the raw source email for mails that don't have a ticket number in the subject.

This setting is not active by default.

Default value:

```
$Self->{'PostMaster::CheckFollowUpModule'}->{'0500-RawEmail'} =  {
  'Module' => 'Kernel::System::PostMaster::FollowUpCheck::RawEmail'
};
```

SendNoAutoResponseRegExp

If this regex matches, no message will be send by the autoresponder.

This setting can not be deactivated.

Default value:

```
$Self->{'SendNoAutoResponseRegExp'} =  '(MAILER-DAEMON|postmaster|abuse)@.+?\\..+?';
```

AutoResponseForWebTickets

If this option is set to 'Yes', tickets created via the web interface, via Customers or Agents, will receive an autoresponse if configured. If this option is set to 'No', no autoresponses will be sent.

This setting can not be deactivated.

Default value:

```
$Self->{'AutoResponseForWebTickets'} =  '1';
```

Ticket → Core::Queue

Queue::EventModulePost###130-UpdateQueue

Event module that performs an update statement on TicketIndex to rename the queue name there if needed and if StaticDB is actually used.

Default value:

```
$Self->{'Queue::EventModulePost'}->{'130-UpdateQueue'} =  {
  'Event' => 'QueueUpdate',
  'Module' => 'Kernel::System::Queue::Event::TicketAcceleratorUpdate',
  'Transaction' => '0'
};
```

Ticket → Core::Stats

Stats::DynamicObjectRegistration###Ticket
Module to generate ticket statistics.

Default value:

```
$Self->{'Stats::DynamicObjectRegistration'}->{'Ticket'} =  {
  'Module' => 'Kernel::System::Stats::Dynamic::Ticket'
};
```

Stats::DynamicObjectRegistration###TicketList
Determines if the statistics module may generate ticket lists.

Default value:

```
$Self->{'Stats::DynamicObjectRegistration'}->{'TicketList'} =  {
  'Module' => 'Kernel::System::Stats::Dynamic::TicketList'
};
```

Stats::DynamicObjectRegistration###TicketAccountedTime
Module to generate accounted time ticket statistics.

Default value:

```
$Self->{'Stats::DynamicObjectRegistration'}->{'TicketAccountedTime'} =  {
  'Module' => 'Kernel::System::Stats::Dynamic::TicketAccountedTime'
};
```

Stats::DynamicObjectRegistration###TicketSolutionResponseTime
Module to generate ticket solution and response time statistics.

Default value:

```
$Self->{'Stats::DynamicObjectRegistration'}->{'TicketSolutionResponseTime'} =  {
  'Module' => 'Kernel::System::Stats::Dynamic::TicketSolutionResponseTime'
};
```

Ticket → Core::Ticket

Ticket::Hook
The identifier for a ticket, e.g. Ticket#, Call#, MyTicket#. The default is Ticket#.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Hook'} =  'Ticket#';
```

Ticket::HookDivider
The divider between TicketHook and ticket number. E.g ': '.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::HookDivider'} =  '';
```

Ticket::SubjectSize
Max size of the subjects in an email reply and in some overview screens.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::SubjectSize'} =  '100';
```

Ticket::SubjectRe

The text at the beginning of the subject in an email reply, e.g. RE, AW, or AS.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::SubjectRe'} =  'Re';
```

Ticket::SubjectFwd

The text at the beginning of the subject when an email is forwarded, e.g. FW, Fwd, or WG.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::SubjectFwd'} =  'Fwd';
```

Ticket::SubjectFormat

The format of the subject. 'Left' means '[TicketHook#:12345] Some Subject', 'Right' means 'Some Subject [TicketHook#:12345]', 'None' means 'Some Subject' and no ticket number. In the latter case you should verify that the setting PostMaster::CheckFollowUpModule###0200-References is activated to recognize followups based on email headers.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::SubjectFormat'} =  'Left';
```

Ticket::MergeDynamicFields

A list of dynamic fields that are merged into the main ticket during a merge operation. Only dynamic fields that are empty in the main ticket will be set.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::MergeDynamicFields'} =  [];
```

Ticket::CustomQueue

Name of custom queue. The custom queue is a queue selection of your preferred queues and can be selected in the preferences settings.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::CustomQueue'} =  'My Queues';
```

Ticket::CustomService

Name of custom service. The custom service is a service selection of your preferred services and can be selected in the preferences settings.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::CustomService'} =  'My Services';
```

Ticket::NewArticleIgnoreSystemSender

Ignore article with system sender type for new article feature (e. g. auto responses or email notifications).

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::NewArticleIgnoreSystemSender'} =  '0';
```

Ticket::ChangeOwnerToEveryone
Changes the owner of tickets to everyone (useful for ASP). Normally only agent with rw permissions in the queue of the ticket will be shown.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::ChangeOwnerToEveryone'} =  '0';
```

Ticket::Responsible
Enables ticket responsible feature, to keep track of a specific ticket.

Default value:

```
$Self->{'Ticket::Responsible'} =  '0';
```

Ticket::ResponsibleAutoSet
Automatically sets the owner of a ticket as the responsible for it (if ticket responsible feature is enabled). This will only work by manually actions of the logged in user. It does not work for automated actions e.g. GenericAgent, Postmaster and GenericInterface.

Default value:

```
$Self->{'Ticket::ResponsibleAutoSet'} =  '1';
```

Ticket::Type
Allows defining new types for ticket (if ticket type feature is enabled).

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Type'} =  '0';
```

Ticket::Type::Default
Defines the default ticket type.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Type::Default'} =  'Unclassified';
```

Ticket::Service
Allows defining services and SLAs for tickets (e. g. email, desktop, network, ...), and escalation attributes for SLAs (if ticket service/SLA feature is enabled).

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Service'} =  '0';
```

Ticket::Service::KeepChildren
Retains all services in listings even if they are children of invalid elements.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Service::KeepChildren'} =  '0';
```

Ticket::Service::Default::UnknownCustomer
Allows default services to be selected also for non existing customers.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Service::Default::UnknownCustomer'} =  '0';
```

Ticket::ArchiveSystem
Activates the ticket archive system to have a faster system by moving some tickets out of the daily scope. To search for these tickets, the archive flag has to be enabled in the ticket search.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::ArchiveSystem'} =  '0';
```

Ticket::ArchiveSystem::RemoveSeenFlags
Controls if the ticket and article seen flags are removed when a ticket is archived.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::ArchiveSystem::RemoveSeenFlags'} =  '1';
```

Ticket::ArchiveSystem::RemoveTicketWatchers
Removes the ticket watcher information when a ticket is archived.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::ArchiveSystem::RemoveTicketWatchers'} =  '1';
```

Ticket::CustomerArchiveSystem
Activates the ticket archive system search in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::CustomerArchiveSystem'} =  '0';
```

Ticket::NumberGenerator
Selects the ticket number generator module. "AutoIncrement" increments the ticket number, the SystemID and the counter are used with SystemID.counter format (e.g. 1010138, 1010139). With "Date" the ticket numbers will be generated by the current date, the SystemID and the counter. The format looks like Year.Month.Day.SystemID.counter (e.g. 200206231010138, 200206231010139). With "DateChecksum" the counter will be appended as checksum to the string of date and SystemID. The checksum will be rotated on a daily basis. The format looks like Year.Month.Day.SystemID.Counter.CheckSum (e.g. 2002070110101520, 2002070110101535). "Random" generates randomized ticket numbers in the format "SystemID.Random" (e.g. 100057866352, 103745394596).

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::NumberGenerator'} =  'Kernel::System::Ticket::Number::DateChecksum';
```

Ticket::NumberGenerator::CheckSystemID

Checks the SystemID in ticket number detection for follow-ups (use "No" if SystemID has been changed after using the system).

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::NumberGenerator::CheckSystemID'} =  '1';
```

Ticket::NumberGenerator::MinCounterSize

Sets the minimal ticket counter size if "AutoIncrement" was selected as TicketNumberGenerator. Default is 5, this means the counter starts from 10000.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::NumberGenerator::MinCounterSize'} =  '5';
```

Ticket::NumberGenerator::Date::UseFormattedCounter

Enables the minimal ticket counter size (if "Date" was selected as TicketNumberGenerator).

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::NumberGenerator::Date::UseFormattedCounter'} =  '0';
```

Ticket::CounterLog

Log file for the ticket counter.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::CounterLog'} =  '<OTRS_CONFIG_Home>/var/log/TicketCounter.log';
```

Ticket::IndexModule

IndexAccelerator: to choose your backend TicketViewAccelerator module. "RuntimeDB" generates each queue view on the fly from ticket table (no performance problems up to approx. 60.000 tickets in total and 6.000 open tickets in the system). "StaticDB" is the most powerful module, it uses an extra ticket-index table that works like a view (recommended if more than 80.000 and 6.000 open tickets are stored in the system). Use the command "bin/otrs.Console.pl Maint::Ticket::QueueIndexRebuild" for initial index creation.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::IndexModule'} =
'Kernel::System::Ticket::IndexAccelerator::RuntimeDB';
```

Ticket::StorageModule

Saves the attachments of articles. "DB" stores all data in the database (not recommended for storing big attachments). "FS" stores the data on the filesystem; this is faster but the webserver should run under the OTRS user. You can switch between the modules even on a system that is already in production without any loss of data. Note: Searching for attachment names is not supported when "FS" is used.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::StorageModule'} =  'Kernel::System::Ticket::ArticleStorageDB';
```

Ticket::StorageModule::CheckAllBackends

Specifies whether all storage backends should be checked when looking for attachments. This is only required for installations where some attachments are in the file system, and others in the database.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::StorageModule::CheckAllBackends'} =  '0';
```

ArticleDir

Specifies the directory to store the data in, if "FS" was selected for TicketStorageModule.

This setting can not be deactivated.

Default value:

```
$Self->{'ArticleDir'} =  '<OTRS_CONFIG_Home>/var/article';
```

OTRSEscalationEvents::DecayTime

The duration in minutes after emitting an event, in which the new escalation notify and start events are suppressed.

Default value:

```
$Self->{'OTRSEscalationEvents::DecayTime'} =  '1440';
```

Ticket::EventModulePost###100-ArchiveRestore

Restores a ticket from the archive (only if the event is a state change to any open available state).

Default value:

```
$Self->{'Ticket::EventModulePost'}->{'100-ArchiveRestore'} =  {
  'Event' => 'TicketStateUpdate',
  'Module' => 'Kernel::System::Ticket::Event::ArchiveRestore'
};
```

Ticket::EventModulePost###110-AcceleratorUpdate

Updates the ticket index accelerator.

Default value:

```
$Self->{'Ticket::EventModulePost'}->{'110-AcceleratorUpdate'} =  {
  'Event' => 'TicketStateUpdate|TicketQueueUpdate|TicketLockUpdate',
  'Module' => 'Kernel::System::Ticket::Event::TicketAcceleratorUpdate'
};
```

Ticket::EventModulePost###120-ForceOwnerResetOnMove

Resets and unlocks the owner of a ticket if it was moved to another queue.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::EventModulePost'}->{'120-ForceOwnerResetOnMove'} =  {
  'Event' => 'TicketQueueUpdate',
  'Module' => 'Kernel::System::Ticket::Event::ForceOwnerReset'
```

```
};
```

Ticket::EventModulePost###130-ForceStateChangeOnLock

Forces to choose a different ticket state (from current) after lock action. Define the current state as key, and the next state after lock action as content.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::EventModulePost'}->{'130-ForceStateChangeOnLock'} =  {
  'Event' => 'TicketLockUpdate',
  'Module' => 'Kernel::System::Ticket::Event::ForceState',
  'new' => 'open'
};
```

Ticket::EventModulePost###140-ResponsibleAutoSet

Automatically sets the responsible of a ticket (if it is not set yet) after the first owner update.

Default value:

```
$Self->{'Ticket::EventModulePost'}->{'140-ResponsibleAutoSet'} =  {
  'Event' => 'TicketOwnerUpdate',
  'Module' => 'Kernel::System::Ticket::Event::ResponsibleAutoSet'
};
```

Ticket::EventModulePost###150-TicketPendingTimeReset

Sets the PendingTime of a ticket to 0 if the state is changed to a non-pending state.

Default value:

```
$Self->{'Ticket::EventModulePost'}->{'150-TicketPendingTimeReset'} =  {
  'Event' => 'TicketStateUpdate',
  'Module' => 'Kernel::System::Ticket::Event::TicketPendingTimeReset'
};
```

Ticket::EventModulePost###500-NotificationEvent

Sends the notifications which are configured in the admin interface under "Notfication (Event)".

Default value:

```
$Self->{'Ticket::EventModulePost'}->{'500-NotificationEvent'} =  {
  'Event' => '',
  'Module' => 'Kernel::System::Ticket::Event::NotificationEvent',
  'Transaction' => '1'
};
```

Ticket::EventModulePost###910-EscalationIndex

Updates the ticket escalation index after a ticket attribute got updated.

Default value:

```
$Self->{'Ticket::EventModulePost'}->{'910-EscalationIndex'} =  {
  'Event' => 'TicketSLAUpdate|TicketQueueUpdate|TicketStateUpdate|TicketCreate|
ArticleCreate',
  'Module' => 'Kernel::System::Ticket::Event::TicketEscalationIndex'
};
```

Ticket::EventModulePost###920-EscalationStopEvents

Ticket event module that triggers the escalation stop events.

Default value:

```
$Self->{'Ticket::EventModulePost'}->{'920-EscalationStopEvents'} =  {
  'Event' => 'TicketSLAUpdate|TicketQueueUpdate|TicketStateUpdate|ArticleCreate',
```

```
  'Module' => 'Kernel::System::Ticket::Event::TriggerEscalationStopEvents'
};
```

Ticket::EventModulePost###930-ForceUnlockOnMove

Forces to unlock tickets after being moved to another queue.

Default value:

```
$Self->{'Ticket::EventModulePost'}->{'930-ForceUnlockOnMove'} =  {
  'Event' => 'TicketQueueUpdate',
  'Module' => 'Kernel::System::Ticket::Event::ForceUnlock'
};
```

Ticket::EventModulePost###940-TicketArticleNewMessageUpdate

Update Ticket "Seen" flag if every article got seen or a new Article got created.

Default value:

```
$Self->{'Ticket::EventModulePost'}->{'940-TicketArticleNewMessageUpdate'} =  {
  'Event' => 'ArticleCreate|ArticleFlagSet',
  'Module' => 'Kernel::System::Ticket::Event::TicketNewMessageUpdate'
};
```

DynamicFieldFromCustomerUser::Mapping

Define a mapping between variables of the customer user data (keys) and dynamic fields of a ticket (values). The purpose is to store customer user data in ticket dynamic fields. The dynamic fields must be present in the system and should be enabled for AgentTicketFreeText, so that they can be set/updated manually by the agent. They mustn't be enabled for AgentTicketPhone, AgentTicketEmail and AgentTicketCustomer. If they were, they would have precedence over the automatically set values. To use this mapping, you have to also activate the next setting below.

This setting is not active by default.

Default value:

```
$Self->{'DynamicFieldFromCustomerUser::Mapping'} =  {
  'UserFirstname' => 'CustomerFirstname'
};
```

Ticket::EventModulePost###950-DynamicFieldFromCustomerUser

This event module stores attributes from CustomerUser as DynamicFields tickets. Please see the setting above for how to configure the mapping.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::EventModulePost'}->{'950-DynamicFieldFromCustomerUser'} =  {
  'Event' => '(TicketCreate|TicketCustomerUpdate)',
  'Module' => 'Kernel::System::Ticket::Event::DynamicFieldFromCustomerUser'
};
```

Ticket::CustomModule###001-CustomModule

Overloads (redefines) existing functions in Kernel::System::Ticket. Used to easily add customizations.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::CustomModule'}->{'001-CustomModule'} =
'Kernel::System::Ticket::Custom';
```

Ticket::ViewableSenderTypes

Defines the default viewable sender types of a ticket (default: customer).

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::ViewableSenderTypes'} =  [
  '\'customer\''
];
```

Ticket::ViewableLocks

Defines the viewable locks of a ticket. NOTE: When you change this setting, make sure to delete the cache in order to use the new value. Default: unlock, tmp_lock.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::ViewableLocks'} =  [
  '\'unlock\'',
  '\'tmp_lock\''
];
```

Ticket::ViewableStateType

Defines the valid state types for a ticket.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::ViewableStateType'} =  [
  'new',
  'open',
  'pending reminder',
  'pending auto'
];
```

Ticket::UnlockStateType

Defines the valid states for unlocked tickets. To unlock tickets the script "bin/otrs.Console.pl Maint::Ticket::UnlockTimeout" can be used.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::UnlockStateType'} =  [
  'new',
  'open'
];
```

Ticket::PendingNotificationOnlyToOwner

Sends reminder notifications of unlocked ticket after reaching the reminder date (only sent to ticket owner).

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::PendingNotificationOnlyToOwner'} =  '0';
```

Ticket::PendingNotificationNotToResponsible

Disables sending reminder notifications to the responsible agent of a ticket (Ticket::Responsible needs to be activated).

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::PendingNotificationNotToResponsible'} =  '0';
```

Ticket::PendingReminderStateType

Defines the state type of the reminder for pending tickets.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::PendingReminderStateType'} =  [
  'pending reminder'
];
```

Ticket::PendingAutoStateType

Determines the possible states for pending tickets that changed state after reaching time limit.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::PendingAutoStateType'} =  [
  'pending auto'
];
```

Ticket::StateAfterPending

Defines which states should be set automatically (Content), after the pending time of state (Key) has been reached.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::StateAfterPending'} =  {
  'pending auto close+' => 'closed successful',
  'pending auto close-' => 'closed unsuccessful'
};
```

System::Permission

Standard available permissions for agents within the application. If more permissions are needed, they can be entered here. Permissions must be defined to be effective. Some other good permissions have also been provided built-in: note, close, pending, customer, freetext, move, compose, responsible, forward, and bounce. Make sure that "rw" is always the last registered permission.

This setting can not be deactivated.

Default value:

```
$Self->{'System::Permission'} =  [
  'ro',
  'move_into',
  'create',
  'note',
  'owner',
  'priority',
  'rw'
];
```

Ticket::Permission###1-OwnerCheck

Module to grant access to the owner of a ticket.

Default value:

```
$Self->{'Ticket::Permission'}->{'1-OwnerCheck'} =  {
  'Granted' => '1',
  'Module' => 'Kernel::System::Ticket::Permission::OwnerCheck',
  'Required' => '0'
};
```

Ticket::Permission::OwnerCheck::Queues

Optional queue limitation for the OwnerCheck permission module. If set, permission is only granted for tickets in the specified queues.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Permission::OwnerCheck::Queues'} =  {
  'Misc' => 'note',
  'Postmaster' => 'ro, move, note',
  'Raw' => 'rw'
};
```

Ticket::Permission###2-ResponsibleCheck

Module to grant access to the agent responsible of a ticket.

Default value:

```
$Self->{'Ticket::Permission'}->{'2-ResponsibleCheck'} =  {
  'Granted' => '1',
  'Module' => 'Kernel::System::Ticket::Permission::ResponsibleCheck',
  'Required' => '0'
};
```

Ticket::Permission::ResponsibleCheck::Queues

Optional queue limitation for the ResponsibleCheck permission module. If set, permission is only granted for tickets in the specified queues.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Permission::ResponsibleCheck::Queues'} =  {
  'Misc' => 'note',
  'Postmaster' => 'ro, move, note',
  'Raw' => 'rw'
};
```

Ticket::Permission###3-GroupCheck

Module to check the group permissions for the access to tickets.

Default value:

```
$Self->{'Ticket::Permission'}->{'3-GroupCheck'} =  {
  'Granted' => '1',
  'Module' => 'Kernel::System::Ticket::Permission::GroupCheck',
  'Required' => '0'
};
```

Ticket::Permission###4-WatcherCheck

Module to grant access to the watcher agents of a ticket.

Default value:

```
$Self->{'Ticket::Permission'}->{'4-WatcherCheck'} =  {
  'Granted' => '1',
  'Module' => 'Kernel::System::Ticket::Permission::WatcherCheck',
  'Required' => '0'
};
```

Ticket::Permission###5-CreatorCheck

Module to grant access to the creator of a ticket.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Permission'}->{'5-CreatorCheck'} =  {
  'Granted' => '1',
  'Module' => 'Kernel::System::Ticket::Permission::CreatorCheck',
  'Required' => '0'
};
```

Ticket::Permission::CreatorCheck::Queues

Optional queue limitation for the CreatorCheck permission module. If set, permission is only granted for tickets in the specified queues.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Permission::CreatorCheck::Queues'} =  {
  'Misc' => 'note',
  'Postmaster' => 'ro, move, note',
  'Raw' => 'rw'
};
```

Ticket::Permission###6-InvolvedCheck

Module to grant access to any agent that has been involved in a ticket in the past (based on ticket history entries).

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Permission'}->{'6-InvolvedCheck'} =  {
  'Granted' => '1',
  'Module' => 'Kernel::System::Ticket::Permission::InvolvedCheck',
  'Required' => '0'
};
```

Ticket::Permission::InvolvedCheck::Queues

Optional queue limitation for the InvolvedCheck permission module. If set, permission is only granted for tickets in the specified queues.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Permission::InvolvedCheck::Queues'} =  {
  'Misc' => 'note',
  'Postmaster' => 'ro, move, note',
  'Raw' => 'rw'
};
```

CustomerTicket::Permission###1-GroupCheck

Module to check the group permissions for customer access to tickets.

Default value:

```
$Self->{'CustomerTicket::Permission'}->{'1-GroupCheck'} =  {
  'Granted' => '0',
  'Module' => 'Kernel::System::Ticket::CustomerPermission::GroupCheck',
  'Required' => '1'
};
```

CustomerTicket::Permission###2-CustomerUserIDCheck

Module to grant access if the CustomerUserID of the ticket matches the CustomerUserID of the customer.

Default value:

```
$Self->{'CustomerTicket::Permission'}->{'2-CustomerUserIDCheck'} =  {
  'Granted' => '1',
```

```
  'Module' => 'Kernel::System::Ticket::CustomerPermission::CustomerUserIDCheck',
  'Required' => '0'
};
```

CustomerTicket::Permission###3-CustomerIDCheck

Module to grant access if the CustomerID of the ticket matches the CustomerID of the customer.

Default value:

```
$Self->{'CustomerTicket::Permission'}->{'3-CustomerIDCheck'} =  {
  'Granted' => '1',
  'Module' => 'Kernel::System::Ticket::CustomerPermission::CustomerIDCheck',
  'Required' => '0'
};
```

Ticket::DefineEmailFrom

Defines how the From field from the emails (sent from answers and email tickets) should look like.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::DefineEmailFrom'} =  'SystemAddressName';
```

Ticket::DefineEmailFromSeparator

Defines the separator between the agents real name and the given queue email address.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::DefineEmailFromSeparator'} =  'via';
```

CustomerNotifyJustToRealCustomer

Sends customer notifications just to the mapped customer.

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerNotifyJustToRealCustomer'} =  '0';
```

AgentSelfNotifyOnAction

Specifies if an agent should receive email notification of his own actions.

This setting can not be deactivated.

Default value:

```
$Self->{'AgentSelfNotifyOnAction'} =  '0';
```

Ticket::EventModulePost###900-GenericAgent

Event module registration. For more performance you can define a trigger event (e. g. Event => TicketCreate).

Default value:

```
$Self->{'Ticket::EventModulePost'}->{'900-GenericAgent'} =  {
  'Event' => '',
  'Module' => 'Kernel::System::Ticket::Event::GenericAgent',
  'Transaction' => '1'
};
```

Ticket::GenericAgentTicketSearch###ExtendedSearchCondition

Allows extended search conditions in ticket search of the generic agent interface. With this feature you can search e. g. ticket title with this kind of conditions like "(*key1*&&*key2*)" or "(*key1*||*key2*)".

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::GenericAgentTicketSearch'}->{'ExtendedSearchCondition'} =  '1';
```

Ticket::GenericAgentRunLimit

Set the limit of tickets that will be executed on a single genericagent job execution.

Default value:

```
$Self->{'Ticket::GenericAgentRunLimit'} =  '4000';
```

Ticket::UnlockOnAway

Unlock tickets whenever a note is added and the owner is out of office.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::UnlockOnAway'} =  '1';
```

Ticket::IncludeUnknownTicketCustomers

Include unknown customers in ticket filter.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::IncludeUnknownTicketCustomers'} =  '0';
```

StandardTemplate::Types

Defines the list of types for templates.

Default value:

```
$Self->{'StandardTemplate::Types'} =  {
  'Answer' => 'Answer',
  'Create' => 'Create',
  'Email' => 'Email',
  'Forward' => 'Forward',
  'Note' => 'Note',
  'PhoneCall' => 'Phone call'
};
```

StandardTemplate2QueueByCreating

List of default Standard Templates which are assigned automatically to new Queues upon creation.

This setting is not active by default.

Default value:

```
$Self->{'StandardTemplate2QueueByCreating'} =  [
  ''
];
```

Ticket::Frontend::DefaultRecipientDisplayType

Default display type for recipient (To,Cc) names in AgentTicketZoom and CustomerTicketZoom.

Default value:

```
$Self->{'Ticket::Frontend::DefaultRecipientDisplayType'} =  'Realname';
```

Ticket::Frontend::DefaultSenderDisplayType

Default display type for sender (From) names in AgentTicketZoom and CustomerTicketZoom.

Default value:

```
$Self->{'Ticket::Frontend::DefaultSenderDisplayType'} =  'Realname';
```

Ticket → Core::TicketACL

Ticket::Acl::Module###1-Ticket::Acl::Module

ACL module that allows closing parent tickets only if all its children are already closed ("State" shows which states are not available for the parent ticket until all child tickets are closed).

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Acl::Module'}->{'1-Ticket::Acl::Module'} =  {
  'Module' => 'Kernel::System::Ticket::Acl::CloseParentAfterClosedChilds',
  'State' => [
    'closed successful',
    'closed unsuccessful'
  ]
};
```

TicketACL::Default::Action

Default ACL values for ticket actions.

This setting can not be deactivated.

Default value:

```
$Self->{'TicketACL::Default::Action'} =  {};
```

ACLKeysLevel1Match

Defines which items are available in first level of the ACL structure.

Default value:

```
$Self->{'ACLKeysLevel1Match'} =  {
  'Properties' => 'Properties',
  'PropertiesDatabase' => 'PropertiesDatabase'
};
```

ACLKeysLevel1Change

Defines which items are available in first level of the ACL structure.

Default value:

```
$Self->{'ACLKeysLevel1Change'} =  {
  'Possible' => 'Possible',
  'PossibleAdd' => 'PossibleAdd',
  'PossibleNot' => 'PossibleNot'
};
```

ACLKeysLevel2::Possible

Defines which items are available in second level of the ACL structure.

Default value:

```
$Self->{'ACLKeysLevel2::Possible'} =  {
  'Action' => 'Action',
  'ActivityDialog' => 'ActivityDialog',
```

```
  'Process' => 'Process',
  'Ticket' => 'Ticket'
};
```

ACLKeysLevel2::PossibleAdd

Defines which items are available in second level of the ACL structure.

Default value:

```
 $Self->{'ACLKeysLevel2::PossibleAdd'} =  {
  'Action' => 'Action',
  'ActivityDialog' => 'ActivityDialog',
  'Process' => 'Process',
  'Ticket' => 'Ticket'
};
```

ACLKeysLevel2::PossibleNot

Defines which items are available in second level of the ACL structure.

Default value:

```
 $Self->{'ACLKeysLevel2::PossibleNot'} =  {
  'Action' => 'Action',
  'ActivityDialog' => 'ActivityDialog',
  'Process' => 'Process',
  'Ticket' => 'Ticket'
};
```

ACLKeysLevel2::Properties

Defines which items are available in second level of the ACL structure.

Default value:

```
 $Self->{'ACLKeysLevel2::Properties'} =  {
  'CustomerUser' => 'CustomerUser',
  'DynamicField' => 'DynamicField',
  'Frontend' => 'Frontend',
  'Owner' => 'Owner',
  'Priority' => 'Priority',
  'Process' => 'Process',
  'Queue' => 'Queue',
  'Responsible' => 'Responsible',
  'SLA' => 'SLA',
  'Service' => 'Service',
  'State' => 'State',
  'Ticket' => 'Ticket',
  'Type' => 'Type',
  'User' => 'User'
};
```

ACLKeysLevel2::PropertiesDatabase

Defines which items are available in second level of the ACL structure.

Default value:

```
 $Self->{'ACLKeysLevel2::PropertiesDatabase'} =  {
  'CustomerUser' => 'CustomerUser',
  'DynamicField' => 'DynamicField',
  'Owner' => 'Owner',
  'Priority' => 'Priority',
  'Process' => 'Process',
  'Queue' => 'Queue',
  'Responsible' => 'Responsible',
  'SLA' => 'SLA',
  'Service' => 'Service',
  'State' => 'State',
  'Ticket' => 'Ticket',
  'Type' => 'Type',
  'User' => 'User'
};
```

ACLKeysLevel3::Actions###100-Default

Defines which items are available for 'Action' in third level of the ACL structure.

Default value:

```
$Self->{'ACLKeysLevel3::Actions'}->{'100-Default'} =  [
 'AgentTicketBounce',
 'AgentTicketClose',
 'AgentTicketCompose',
 'AgentTicketCustomer',
 'AgentTicketForward',
 'AgentTicketEmailOutbound',
 'AgentTicketFreeText',
 'AgentTicketHistory',
 'AgentTicketLink',
 'AgentTicketLock',
 'AgentTicketMerge',
 'AgentTicketMove',
 'AgentTicketNote',
 'AgentTicketOwner',
 'AgentTicketPending',
 'AgentTicketPhone',
 'AgentTicketPhoneInbound',
 'AgentTicketPhoneOutbound',
 'AgentTicketPlain',
 'AgentTicketPrint',
 'AgentTicketPriority',
 'AgentTicketProcess',
 'AgentTicketResponsible',
 'AgentTicketSearch',
 'AgentTicketWatcher',
 'AgentTicketZoom',
 'AgentLinkObject',
 'CustomerTicketProcess'
];
```

ACL::CacheTTL

Cache time in seconds for the DB ACL backend.

This setting can not be deactivated.

Default value:

```
$Self->{'ACL::CacheTTL'} =  '3600';
```

TicketACL::Debug::Enabled

If enabled debugging information for ACLs is logged.

This setting can not be deactivated.

Default value:

```
$Self->{'TicketACL::Debug::Enabled'} =  '0';
```

TicketACL::Debug::LogPriority

Defines the priority in which the information is logged and presented.

This setting is not active by default.

Default value:

```
$Self->{'TicketACL::Debug::LogPriority'} =  'debug';
```

TicketACL::Debug::Filter###00-Default

Filter for debugging ACLs. Note: More ticket attributes can be added in the format <OTRS_TICKET_Attribute> e.g. <OTRS_TICKET_Priority>.

This setting is not active by default.

Default value:

```
$Self->{'TicketACL::Debug::Filter'}->{'00-Default'} =  {
  '<OTRS_TICKET_TicketNumber>' => '',
  'ACLName' => ''
};
```

Ticket → Core::TicketBulkAction

Ticket::Frontend::BulkFeature

Enables ticket bulk action feature for the agent frontend to work on more than one ticket at a time.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::BulkFeature'} =  '1';
```

Ticket::Frontend::BulkFeatureGroup

Enables ticket bulk action feature only for the listed groups.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::BulkFeatureGroup'} =  [
  'admin',
  'users'
];
```

Ticket → Core::TicketDynamicFieldDefault

Ticket::EventModulePost###TicketDynamicFieldDefault

Event module registration. For more performance you can define a trigger event (e. g. Event => TicketCreate). This is only possible if all Ticket dynamic fields need the same event.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::EventModulePost'}->{'TicketDynamicFieldDefault'} =  {
  'Module' => 'Kernel::System::Ticket::Event::TicketDynamicFieldDefault',
  'Transaction' => '1'
};
```

Ticket::TicketDynamicFieldDefault###Element1

Configures a default TicketDynamicField setting. "Name" defines the dynamic field which should be used, "Value" is the data that will be set, and "Event" defines the trigger event. Please check the developer manual (http://otrs.github.io/doc/), chapter "Ticket Event Module".

This setting is not active by default.

Default value:

```
$Self->{'Ticket::TicketDynamicFieldDefault'}->{'Element1'} =  {
  'Event' => 'TicketCreate',
  'Name' => 'Field1',
  'Value' => 'Default'
};
```

Ticket::TicketDynamicFieldDefault###Element2

Configures a default TicketDynamicField setting. "Name" defines the dynamic field which should be used, "Value" is the data that will be set, and "Event" defines the

trigger event. Please check the developer manual (http://otrs.github.io/doc/), chapter "Ticket Event Module".

This setting is not active by default.

Default value:

```
$Self->{'Ticket::TicketDynamicFieldDefault'}->{'Element2'} =  {
  'Event' => '',
  'Name' => '',
  'Value' => ''
};
```

Ticket::TicketDynamicFieldDefault###Element3

Configures a default TicketDynamicField setting. "Name" defines the dynamic field which should be used, "Value" is the data that will be set, and "Event" defines the trigger event. Please check the developer manual (http://otrs.github.io/doc/), chapter "Ticket Event Module".

This setting is not active by default.

Default value:

```
$Self->{'Ticket::TicketDynamicFieldDefault'}->{'Element3'} =  {
  'Event' => '',
  'Name' => '',
  'Value' => ''
};
```

Ticket::TicketDynamicFieldDefault###Element4

Configures a default TicketDynamicField setting. "Name" defines the dynamic field which should be used, "Value" is the data that will be set, and "Event" defines the trigger event. Please check the developer manual (http://otrs.github.io/doc/), chapter "Ticket Event Module".

This setting is not active by default.

Default value:

```
$Self->{'Ticket::TicketDynamicFieldDefault'}->{'Element4'} =  {
  'Event' => '',
  'Name' => '',
  'Value' => ''
};
```

Ticket::TicketDynamicFieldDefault###Element5

Configures a default TicketDynamicField setting. "Name" defines the dynamic field which should be used, "Value" is the data that will be set, and "Event" defines the trigger event. Please check the developer manual (http://otrs.github.io/doc/), chapter "Ticket Event Module".

This setting is not active by default.

Default value:

```
$Self->{'Ticket::TicketDynamicFieldDefault'}->{'Element5'} =  {
  'Event' => '',
  'Name' => '',
  'Value' => ''
};
```

Ticket::TicketDynamicFieldDefault###Element6

Configures a default TicketDynamicField setting. "Name" defines the dynamic field which should be used, "Value" is the data that will be set, and "Event" defines the trigger event. Please check the developer manual (http://otrs.github.io/doc/), chapter "Ticket Event Module".

This setting is not active by default.

Default value:

```
$Self->{'Ticket::TicketDynamicFieldDefault'}->{'Element6'} =  {
  'Event' => '',
  'Name' => '',
  'Value' => ''
};
```

Ticket::TicketDynamicFieldDefault###Element7

Configures a default TicketDynamicField setting. "Name" defines the dynamic field which should be used, "Value" is the data that will be set, and "Event" defines the trigger event. Please check the developer manual (http://otrs.github.io/doc/), chapter "Ticket Event Module".

This setting is not active by default.

Default value:

```
$Self->{'Ticket::TicketDynamicFieldDefault'}->{'Element7'} =  {
  'Event' => '',
  'Name' => '',
  'Value' => ''
};
```

Ticket::TicketDynamicFieldDefault###Element8

Configures a default TicketDynamicField setting. "Name" defines the dynamic field which should be used, "Value" is the data that will be set, and "Event" defines the trigger event. Please check the developer manual (http://otrs.github.io/doc/), chapter "Ticket Event Module".

This setting is not active by default.

Default value:

```
$Self->{'Ticket::TicketDynamicFieldDefault'}->{'Element8'} =  {
  'Event' => '',
  'Name' => '',
  'Value' => ''
};
```

Ticket::TicketDynamicFieldDefault###Element9

Configures a default TicketDynamicField setting. "Name" defines the dynamic field which should be used, "Value" is the data that will be set, and "Event" defines the trigger event. Please check the developer manual (http://otrs.github.io/doc/), chapter "Ticket Event Module".

This setting is not active by default.

Default value:

```
$Self->{'Ticket::TicketDynamicFieldDefault'}->{'Element9'} =  {
  'Event' => '',
  'Name' => '',
  'Value' => ''
};
```

Ticket::TicketDynamicFieldDefault###Element10

Configures a default TicketDynamicField setting. "Name" defines the dynamic field which should be used, "Value" is the data that will be set, and "Event" defines the trigger event. Please check the developer manual (http://otrs.github.io/doc/), chapter "Ticket Event Module".

This setting is not active by default.

Default value:

```
$Self->{'Ticket::TicketDynamicFieldDefault'}->{'Element10'} =  {
  'Event' => '',
  'Name' => '',
  'Value' => ''
};
```

Ticket::TicketDynamicFieldDefault###Element11

Configures a default TicketDynamicField setting. "Name" defines the dynamic field which should be used, "Value" is the data that will be set, and "Event" defines the trigger event. Please check the developer manual (http://otrs.github.io/doc/), chapter "Ticket Event Module".

This setting is not active by default.

Default value:

```
$Self->{'Ticket::TicketDynamicFieldDefault'}->{'Element11'} =  {
  'Event' => '',
  'Name' => '',
  'Value' => ''
};
```

Ticket::TicketDynamicFieldDefault###Element12

Configures a default TicketDynamicField setting. "Name" defines the dynamic field which should be used, "Value" is the data that will be set, and "Event" defines the trigger event. Please check the developer manual (http://otrs.github.io/doc/), chapter "Ticket Event Module".

This setting is not active by default.

Default value:

```
$Self->{'Ticket::TicketDynamicFieldDefault'}->{'Element12'} =  {
  'Event' => '',
  'Name' => '',
  'Value' => ''
};
```

Ticket::TicketDynamicFieldDefault###Element13

Configures a default TicketDynamicField setting. "Name" defines the dynamic field which should be used, "Value" is the data that will be set, and "Event" defines the trigger event. Please check the developer manual (http://otrs.github.io/doc/), chapter "Ticket Event Module".

This setting is not active by default.

Default value:

```
$Self->{'Ticket::TicketDynamicFieldDefault'}->{'Element13'} =  {
  'Event' => '',
  'Name' => '',
  'Value' => ''
};
```

Ticket::TicketDynamicFieldDefault###Element14

Configures a default TicketDynamicField setting. "Name" defines the dynamic field which should be used, "Value" is the data that will be set, and "Event" defines the trigger event. Please check the developer manual (http://otrs.github.io/doc/), chapter "Ticket Event Module".

This setting is not active by default.

Default value:

```
$Self->{'Ticket::TicketDynamicFieldDefault'}->{'Element14'} =  {
  'Event' => '',
  'Name' => '',
  'Value' => ''
};
```

Ticket::TicketDynamicFieldDefault###Element15

Configures a default TicketDynamicField setting. "Name" defines the dynamic field which should be used, "Value" is the data that will be set, and "Event" defines the trigger event. Please check the developer manual (http://otrs.github.io/doc/), chapter "Ticket Event Module".

This setting is not active by default.

Default value:

```
$Self->{'Ticket::TicketDynamicFieldDefault'}->{'Element15'} =  {
  'Event' => '',
  'Name' => '',
  'Value' => ''
};
```

Ticket::TicketDynamicFieldDefault###Element16

Configures a default TicketDynamicField setting. "Name" defines the dynamic field which should be used, "Value" is the data that will be set, and "Event" defines the trigger event. Please check the developer manual (http://otrs.github.io/doc/), chapter "Ticket Event Module".

This setting is not active by default.

Default value:

```
$Self->{'Ticket::TicketDynamicFieldDefault'}->{'Element16'} =  {
  'Event' => '',
  'Name' => '',
  'Value' => ''
};
```

Ticket → Core::TicketWatcher

Ticket::Watcher

Enables or disables the ticket watcher feature, to keep track of tickets without being the owner nor the responsible.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Watcher'} =  '0';
```

Ticket::WatcherGroup

Enables ticket watcher feature only for the listed groups.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::WatcherGroup'} =  [
  'admin',
  'users'
];
```

Ticket → Frontend::Admin

Events###Ticket

List of all ticket events to be displayed in the GUI.

This setting can not be deactivated.

Default value:

```
$Self->{'Events'}->{'Ticket'} =  [
  'TicketCreate',
  'TicketDelete',
  'TicketTitleUpdate',
  'TicketUnlockTimeoutUpdate',
  'TicketQueueUpdate',
  'TicketTypeUpdate',
  'TicketServiceUpdate',
  'TicketSLAUpdate',
  'TicketCustomerUpdate',
  'TicketPendingTimeUpdate',
  'TicketLockUpdate',
  'TicketArchiveFlagUpdate',
  'TicketStateUpdate',
  'TicketOwnerUpdate',
  'TicketResponsibleUpdate',
  'TicketPriorityUpdate',
  'HistoryAdd',
  'HistoryDelete',
  'TicketAccountTime',
  'TicketMerge',
  'TicketSubscribe',
  'TicketUnsubscribe',
  'TicketFlagSet',
  'TicketFlagDelete',
  'TicketSlaveLinkAdd',
  'TicketSlaveLinkDelete',
  'TicketMasterLinkDelete',
  'EscalationResponseTimeNotifyBefore',
  'EscalationUpdateTimeNotifyBefore',
  'EscalationSolutionTimeNotifyBefore',
  'EscalationResponseTimeStart',
  'EscalationUpdateTimeStart',
  'EscalationSolutionTimeStart',
  'EscalationResponseTimeStop',
  'EscalationUpdateTimeStop',
  'EscalationSolutionTimeStop',
  'NotificationNewTicket',
  'NotificationFollowUp',
  'NotificationLockTimeout',
  'NotificationOwnerUpdate',
  'NotificationResponsibleUpdate',
  'NotificationAddNote',
  'NotificationMove',
  'NotificationPendingReminder',
  'NotificationEscalation',
  'NotificationEscalationNotifyBefore',
  'NotificationServiceUpdate'
];
```

Events###Article

List of all article events to be displayed in the GUI.

This setting can not be deactivated.

Default value:

```
$Self->{'Events'}->{'Article'} =  [
  'ArticleCreate',
  'ArticleUpdate',
  'ArticleSend',
  'ArticleBounce',
  'ArticleAgentNotification',
  'ArticleCustomerNotification',
  'ArticleAutoResponse',
  'ArticleFlagSet',
  'ArticleFlagDelete',
```

```
  'ArticleAgentNotification',
  'ArticleCustomerNotification'
];
```

Events###Queue
List of all queue events to be displayed in the GUI.

This setting can not be deactivated.

Default value:

```
$Self->{'Events'}->{'Queue'} =  [
  'QueueCreate',
  'QueueUpdate'
];
```

Ticket → Frontend::Admin::AdminNotificationEvent

Frontend::Admin::AdminNotificationEvent###RichText
Uses richtext for viewing and editing ticket notification.

Default value:

```
$Self->{'Frontend::Admin::AdminNotificationEvent'}->{'RichText'} =  '1';
```

Frontend::Admin::AdminNotificationEvent###RichTextWidth
Defines the width for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Frontend::Admin::AdminNotificationEvent'}->{'RichTextWidth'} =  '620';
```

Frontend::Admin::AdminNotificationEvent###RichTextHeight
Defines the height for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Frontend::Admin::AdminNotificationEvent'}->{'RichTextHeight'} =  '320';
```

Notification::Transport###Email
Defines all the parameters for this notification transport.

Default value:

```
$Self->{'Notification::Transport'}->{'Email'} =  {
  'AgentEnabledByDefault' => '1',
  'Icon' => 'fa fa-envelope',
  'IsOTRSBusinessTransport' => '0',
  'Module' => 'Kernel::System::Ticket::Event::NotificationEvent::Transport::Email',
  'Name' => 'Email',
  'Prio' => '100'
};
```

Notification::Transport###NotificationView
Defines all the parameters for this notification transport.

Default value:

```
$Self->{'Notification::Transport'}->{'NotificationView'} =  {
  'AgentEnabledByDefault' => '0',
  'Icon' => 'fa fa-th-list',
  'IsOTRSBusinessTransport' => '1',
  'Module' =>
 'Kernel::System::Ticket::Event::NotificationEvent::Transport::NotificationView',
  'Name' => 'Web View',
```

```
  'Prio' => '110'
};
```

Notification::Transport###SMS
Defines all the parameters for this notification transport.

Default value:

```
$Self->{'Notification::Transport'}->{'SMS'} =  {
  'AgentEnabledByDefault' => '0',
  'Icon' => 'fa fa-mobile',
  'IsOTRSBusinessTransport' => '1',
  'Module' => 'Kernel::System::Ticket::Event::NotificationEvent::Transport::SMS',
  'Name' => 'SMS (Short Message Service)',
  'Prio' => '120'
};
```

Notification::CharactersPerLine
Defines the number of character per line used in case an HTML article preview replacement on TemplateGenerator for EventNotifications.

This setting can not be deactivated.

Default value:

```
$Self->{'Notification::CharactersPerLine'} =  '80';
```

Ticket → Frontend::Admin::ModuleRegistration

Frontend::Module###AdminACL
Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminACL'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.ACL.css'
    ],
    'JavaScript' => [
      'Core.Agent.Admin.ACL.js'
    ]
  },
  'NavBarModule' => {
    'Block' => 'Ticket',
    'Description' => 'Configure and manage ACLs.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Access Control Lists (ACL)',
    'Prio' => '750'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Access Control Lists (ACL)'
};
```

Frontend::Module###AdminQueue
Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminQueue'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
```

```
    'Block' => 'Queue',
    'Description' => 'Create and manage queues.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Queues',
    'Prio' => '100'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Queues'
};
```

Frontend::Module###AdminTemplate

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminTemplate'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Queue',
    'Description' => 'Create and manage templates.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Templates',
    'Prio' => '200'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Templates'
};
```

Frontend::Module###AdminQueueTemplates

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminQueueTemplates'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Queue',
    'Description' => 'Link templates to queues.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Templates <-> Queues',
    'Prio' => '300'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Templates <-> Queues'
};
```

Frontend::Module###AdminAutoResponse

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminAutoResponse'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Queue',
    'Description' => 'Create and manage responses that are automatically sent.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Auto Responses',
    'Prio' => '400'
  },
  'NavBarName' => 'Admin',
```

```
  'Title' => 'Auto Responses'
};
```

Frontend::Module###AdminQueueAutoResponse
Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminQueueAutoResponse'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Queue',
    'Description' => 'Link queues to auto responses.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Auto Responses <-> Queues',
    'Prio' => '500'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Auto Responses <-> Queues'
};
```

Frontend::Module###AdminAttachment
Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminAttachment'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Queue',
    'Description' => 'Create and manage attachments.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Attachments',
    'Prio' => '600'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Attachments'
};
```

Frontend::Module###AdminTemplateAttachment
Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminTemplateAttachment'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Queue',
    'Description' => 'Link attachments to templates.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Attachments <-> Templates',
    'Prio' => '700'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Attachments <-> Templates'
};
```

Frontend::Module###AdminSalutation
Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminSalutation'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Queue',
    'Description' => 'Create and manage salutations.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Salutations',
    'Prio' => '800'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Salutations'
};
```

Frontend::Module###AdminSignature

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminSignature'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Queue',
    'Description' => 'Create and manage signatures.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Signatures',
    'Prio' => '900'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Signatures'
};
```

Frontend::Module###AdminSystemAddress

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminSystemAddress'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Email',
    'Description' => 'Set sender email addresses for this system.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Email Addresses',
    'Prio' => '300'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Email Addresses'
};
```

Frontend::Module###AdminNotificationEvent

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminNotificationEvent'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'CSS' => [
      'Core.Agent.Admin.NotificationEvent.css'
```

```
    ],
    'JavaScript' => [
      'Core.Agent.Admin.NotificationEvent.js'
    ]
  },
  'NavBarModule' => {
    'Block' => 'Ticket',
    'Description' => 'Create and manage ticket notifications.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Ticket Notifications',
    'Prio' => '400'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Ticket Notifications'
};
```

Frontend::Module###AdminService

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminService'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Ticket',
    'Description' => 'Create and manage services.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Services',
    'Prio' => '900'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Services'
};
```

Frontend::Module###AdminSLA

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminSLA'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Ticket',
    'Description' => 'Create and manage Service Level Agreements (SLAs).',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Service Level Agreements',
    'Prio' => '1000'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Service Level Agreements'
};
```

Frontend::Module###AdminType

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminType'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Ticket',
    'Description' => 'Create and manage ticket types.',
```

```
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Types',
    'Prio' => '700'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Types'
};
```

Frontend::Module###AdminState

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminState'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Ticket',
    'Description' => 'Create and manage ticket states.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'States',
    'Prio' => '800'
  },
  'NavBarName' => 'Admin',
  'Title' => 'States'
};
```

Frontend::Module###AdminPriority

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminPriority'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'NavBarModule' => {
    'Block' => 'Ticket',
    'Description' => 'Create and manage ticket priorities.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'Priorities',
    'Prio' => '850'
  },
  'NavBarName' => 'Admin',
  'Title' => 'Priorities'
};
```

Frontend::Module###AdminGenericAgent

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AdminGenericAgent'} =  {
  'Description' => 'This module is part of the admin area of OTRS.',
  'Group' => [
    'admin'
  ],
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.Admin.GenericAgent.js'
    ]
  },
  'NavBarModule' => {
    'Block' => 'System',
    'Description' => 'Manage tasks triggered by event or time based execution.',
    'Module' => 'Kernel::Output::HTML::NavBar::ModuleAdmin',
    'Name' => 'GenericAgent',
    'Prio' => '300'
```

```
  },
  'NavBarName' => 'Admin',
  'Title' => 'GenericAgent'
};
```

Ticket → Frontend::Agent

Ticket::Frontend::PendingDiffTime
Time in seconds that gets added to the actual time if setting a pending-state (default: 86400 = 1 day).

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::PendingDiffTime'} =  '86400';
```

Ticket::Frontend::MaxQueueLevel
Define the max depth of queues.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::MaxQueueLevel'} =  '5';
```

Ticket::Frontend::ListType
Shows existing parent/child queue lists in the system in the form of a tree or a list.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::ListType'} =  'tree';
```

Ticket::Frontend::TextAreaEmail
Permitted width for compose email windows.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::TextAreaEmail'} =  '82';
```

Ticket::Frontend::TextAreaNote
Permitted width for compose note windows.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::TextAreaNote'} =  '78';
```

Ticket::Frontend::InformAgentMaxSize
Max size (in rows) of the informed agents box in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::InformAgentMaxSize'} =  '3';
```

Ticket::Frontend::InvolvedAgentMaxSize
Max size (in rows) of the involved agents box in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::InvolvedAgentMaxSize'} =  '3';
```

Ticket::Frontend::CustomerInfoCompose
Shows the customer user information (phone and email) in the compose screen.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerInfoCompose'} =  '1';
```

Ticket::Frontend::CustomerInfoComposeMaxSize
Max size (in characters) of the customer information table (phone and email) in the compose screen.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerInfoComposeMaxSize'} =  '22';
```

Ticket::Frontend::CustomerInfoZoom
Shows the customer user's info in the ticket zoom view.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerInfoZoom'} =  '1';
```

Ticket::Frontend::CustomerInfoZoomMaxSize
Maximum size (in characters) of the customer information table in the ticket zoom view.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerInfoZoomMaxSize'} =  '22';
```

Ticket::Frontend::DynamicFieldsZoomMaxSizeSidebar
Maximum length (in characters) of the dynamic field in the sidebar of the ticket zoom view.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::DynamicFieldsZoomMaxSizeSidebar'} =  '18';
```

Ticket::Frontend::DynamicFieldsZoomMaxSizeArticle
Maximum length (in characters) of the dynamic field in the article of the ticket zoom view.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::DynamicFieldsZoomMaxSizeArticle'} =  '160';
```

Ticket::Frontend::AccountTime
Activates time accounting.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AccountTime'} =  '1';
```

Ticket::Frontend::TimeUnits
Sets the prefered time units (e.g. work units, hours, minutes).

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::TimeUnits'} =  ' (work units)';
```

Ticket::Frontend::NeedAccountedTime
Defines if time accounting is mandatory in the agent interface. If activated, a note must be entered for all ticket actions (no matter if the note itself is configured as active or is originally mandatory for the individual ticket action screen).

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::NeedAccountedTime'} =  '0';
```

Ticket::Frontend::BulkAccountedTime
Defines if time accounting must be set to all tickets in bulk action.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::BulkAccountedTime'} =  '1';
```

Ticket::Frontend::NeedSpellCheck
Defines if composed messages have to be spell checked in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::NeedSpellCheck'} =  '0';
```

Ticket::Frontend::NewOwnerSelection
Shows an owner selection in phone and email tickets in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::NewOwnerSelection'} =  '1';
```

Ticket::Frontend::NewResponsibleSelection
Show a responsible selection in phone and email tickets in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::NewResponsibleSelection'} =  '1';
```

Ticket::Frontend::NewQueueSelectionType
Defines the recipient target of the phone ticket and the sender of the email ticket ("Queue" shows all queues, "System address" displays all system addresses) in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::NewQueueSelectionType'} =  'Queue';
```

Ticket::Frontend::NewQueueSelectionString

Determines the strings that will be shown as recipient (To:) of the phone ticket and as sender (From:) of the email ticket in the agent interface. For Queue as NewQueueSelectionType "<Queue>" shows the names of the queues and for SystemAddress "<Realname> <<Email>>" shows the name and email of the recipient.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::NewQueueSelectionString'} =  '<Queue>';
```

Ticket::Frontend::NewQueueOwnSelection

Determines which options will be valid of the recipient (phone ticket) and the sender (email ticket) in the agent interface.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::NewQueueOwnSelection'} =  {
  '1' => 'First Queue',
  '2' => 'Second Queue'
};
```

Ticket::Frontend::ShowCustomerTickets

Shows customer history tickets in AgentTicketPhone, AgentTicketEmail and AgentTicketCustomer.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::ShowCustomerTickets'} =  '1';
```

NewTicketInNewWindow::Enabled

If enabled, TicketPhone and TicketEmail will be open in new windows.

This setting can not be deactivated.

Default value:

```
$Self->{'NewTicketInNewWindow::Enabled'} =  '0';
```

CustomerDBLink

Defines an external link to the database of the customer (e.g. 'http://yourhost/customer.php?CID=[% Data.CustomerID %]' or '').

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerDBLink'} =  '[% Env("CGIHandle") %]?
Action=AgentCustomerInformationCenter;CustomerID=[% Data.CustomerID | uri %]';
```

CustomerDBLinkTarget

Defines the target attribute in the link to external customer database. E.g. 'target="cdb"'.

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerDBLinkTarget'} =  '';
```

CustomerDBLinkClass

Defines the target attribute in the link to external customer database. E.g. 'AsPopup PopupType_TicketAction'.

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerDBLinkClass'} =  '';
```

Frontend::CommonParam###Action

Defines the default used Frontend-Module if no Action parameter given in the url on the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::CommonParam'}->{'Action'} =  'AgentDashboard';
```

Frontend::CommonParam###QueueID

Default queue ID used by the system in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::CommonParam'}->{'QueueID'} =  '0';
```

Frontend::CommonParam###TicketID

Default ticket ID used by the system in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Frontend::CommonParam'}->{'TicketID'} =  '';
```

DefaultOverviewColumns

General ticket data shown in the ticket overviews (fall-back). Possible settings: 0 = Disabled, 1 = Available, 2 = Enabled by default. Note that TicketNumber can not be disabled, because it is necessary.

This setting can not be deactivated.

Default value:

```
$Self->{'DefaultOverviewColumns'} =  {
 'Age' => '2',
 'Changed' => '1',
 'CustomerID' => '2',
 'CustomerName' => '1',
 'CustomerUserID' => '1',
 'EscalationResponseTime' => '1',
 'EscalationSolutionTime' => '1',
 'EscalationTime' => '1',
 'EscalationUpdateTime' => '1',
 'Lock' => '2',
 'Owner' => '2',
 'PendingTime' => '1',
 'Priority' => '1',
 'Queue' => '2',
 'Responsible' => '1',
 'SLA' => '1',
 'Service' => '1',
 'State' => '2',
```

```
  'TicketNumber' => '2',
  'Title' => '2',
  'Type' => '1'
};
```

Ticket → Frontend::Agent::Dashboard

DashboardBackend###0100-TicketPendingReminder

Parameters for the dashboard backend of the ticket pending reminder overview of the agent interface . "Limit" is the number of entries shown by default. "Group" is used to restrict the access to the plugin (e. g. Group: admin;group1;group2;). "Default" determines if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTLLocal" is the cache time in minutes for the plugin. Note: Only Ticket attributes and Dynamic Fields (DynamicField_NameX) are allowed for DefaultColumns. Possible settings: 0 = Disabled, 1 = Available, 2 = Enabled by default.

Default value:

```
 $Self->{'DashboardBackend'}->{'0100-TicketPendingReminder'} =  {
  'Attributes' => 'TicketPendingTimeOlderMinutes=1;StateType=pending
 reminder;SortBy=PendingTime;OrderBy=Down;',
  'Block' => 'ContentLarge',
  'CacheTTLLocal' => '0.5',
  'Default' => '1',
  'DefaultColumns' => {
    'Age' => '2',
    'Changed' => '1',
    'Created' => '1',
    'CustomerCompanyName' => '1',
    'CustomerID' => '1',
    'CustomerName' => '1',
    'CustomerUserID' => '1',
    'EscalationResponseTime' => '1',
    'EscalationSolutionTime' => '1',
    'EscalationTime' => '1',
    'EscalationUpdateTime' => '1',
    'Lock' => '1',
    'Owner' => '1',
    'PendingTime' => '1',
    'Priority' => '1',
    'Queue' => '1',
    'Responsible' => '1',
    'SLA' => '1',
    'Service' => '1',
    'State' => '1',
    'TicketNumber' => '2',
    'Title' => '2',
    'Type' => '1'
  },
  'Description' => 'All tickets with a reminder set where the reminder date has been
 reached',
  'Filter' => 'Locked',
  'Group' => '',
  'Limit' => '10',
  'Module' => 'Kernel::Output::HTML::Dashboard::TicketGeneric',
  'Permission' => 'rw',
  'Time' => 'UntilTime',
  'Title' => 'Reminder Tickets'
};
```

DashboardBackend###0110-TicketEscalation

Parameters for the dashboard backend of the ticket escalation overview of the agent interface . "Limit" is the number of entries shown by default. "Group" is used to restrict the access to the plugin (e. g. Group: admin;group1;group2;). "Default" determines if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTLLocal" is the cache time in minutes for the plugin. Note: Only Ticket attributes and Dynamic Fields (DynamicField_NameX) are allowed for DefaultColumns. Possible settings: 0 = Disabled, 1 = Available, 2 = Enabled by default.

Default value:

```
$Self->{'DashboardBackend'}->{'0110-TicketEscalation'} =  {
 'Attributes' =>
'TicketEscalationTimeOlderMinutes=1;SortBy=EscalationTime;OrderBy=Down;',
 'Block' => 'ContentLarge',
 'CacheTTLLocal' => '0.5',
 'Default' => '1',
 'DefaultColumns' => {
   'Age' => '2',
   'Changed' => '1',
   'Created' => '1',
   'CustomerCompanyName' => '1',
   'CustomerID' => '1',
   'CustomerName' => '1',
   'CustomerUserID' => '1',
   'EscalationResponseTime' => '1',
   'EscalationSolutionTime' => '1',
   'EscalationTime' => '1',
   'EscalationUpdateTime' => '1',
   'Lock' => '1',
   'Owner' => '1',
   'PendingTime' => '1',
   'Priority' => '1',
   'Queue' => '1',
   'Responsible' => '1',
   'SLA' => '1',
   'Service' => '1',
   'State' => '1',
   'TicketNumber' => '2',
   'Title' => '2',
   'Type' => '1'
 },
 'Description' => 'All escalated tickets',
 'Filter' => 'All',
 'Group' => '',
 'Limit' => '10',
 'Module' => 'Kernel::Output::HTML::Dashboard::TicketGeneric',
 'Permission' => 'rw',
 'Time' => 'EscalationTime',
 'Title' => 'Escalated Tickets'
};
```

DashboardBackend###0120-TicketNew

Parameters for the dashboard backend of the new tickets overview of the agent interface. "Limit" is the number of entries shown by default. "Group" is used to restrict the access to the plugin (e. g. Group: admin;group1;group2;). "Default" determines if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTLLocal" is the cache time in minutes for the plugin. Note: Only Ticket attributes and Dynamic Fields (DynamicField_NameX) are allowed for DefaultColumns. Possible settings: 0 = Disabled, 1 = Available, 2 = Enabled by default.

Default value:

```
$Self->{'DashboardBackend'}->{'0120-TicketNew'} =  {
 'Attributes' => 'StateType=new;',
 'Block' => 'ContentLarge',
 'CacheTTLLocal' => '0.5',
 'Default' => '1',
 'DefaultColumns' => {
   'Age' => '2',
   'Changed' => '1',
   'Created' => '1',
   'CustomerCompanyName' => '1',
   'CustomerID' => '1',
   'CustomerName' => '1',
   'CustomerUserID' => '1',
   'EscalationResponseTime' => '1',
   'EscalationSolutionTime' => '1',
   'EscalationTime' => '1',
```

```
    'EscalationUpdateTime' => '1',
    'Lock' => '1',
    'Owner' => '1',
    'PendingTime' => '1',
    'Priority' => '1',
    'Queue' => '1',
    'Responsible' => '1',
    'SLA' => '1',
    'Service' => '1',
    'State' => '1',
    'TicketNumber' => '2',
    'Title' => '2',
    'Type' => '1'
  },
  'Description' => 'All new tickets, these tickets have not been worked on yet',
  'Filter' => 'All',
  'Group' => '',
  'Limit' => '10',
  'Module' => 'Kernel::Output::HTML::Dashboard::TicketGeneric',
  'Permission' => 'rw',
  'Time' => 'Age',
  'Title' => 'New Tickets'
};
```

DashboardBackend###0130-TicketOpen

Parameters for the dashboard backend of the open tickets overview of the agent interface. "Limit" is the number of entries shown by default. "Group" is used to restrict the access to the plugin (e. g. Group: admin;group1;group2;). "Default" determines if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTLLocal" is the cache time in minutes for the plugin. Note: Only Ticket attributes and Dynamic Fields (DynamicField_NameX) are allowed for DefaultColumns. Possible settings: 0 = Disabled, 1 = Available, 2 = Enabled by default.

Default value:

```
$Self->{'DashboardBackend'}->{'0130-TicketOpen'} =  {
  'Attributes' => 'StateType=open;',
  'Block' => 'ContentLarge',
  'CacheTTLLocal' => '0.5',
  'Default' => '1',
  'DefaultColumns' => {
    'Age' => '2',
    'Changed' => '1',
    'Created' => '1',
    'CustomerCompanyName' => '1',
    'CustomerID' => '1',
    'CustomerName' => '1',
    'CustomerUserID' => '1',
    'EscalationResponseTime' => '1',
    'EscalationSolutionTime' => '1',
    'EscalationTime' => '1',
    'EscalationUpdateTime' => '1',
    'Lock' => '1',
    'Owner' => '1',
    'PendingTime' => '1',
    'Priority' => '1',
    'Queue' => '1',
    'Responsible' => '1',
    'SLA' => '1',
    'Service' => '1',
    'State' => '1',
    'TicketNumber' => '2',
    'Title' => '2',
    'Type' => '1'
  },
  'Description' => 'All open tickets, these tickets have already been worked on, but
 need a response',
  'Filter' => 'All',
  'Group' => '',
  'Limit' => '10',
```

```
  'Module' => 'Kernel::Output::HTML::Dashboard::TicketGeneric',
  'Permission' => 'rw',
  'Time' => 'Age',
  'Title' => 'Open Tickets / Need to be answered'
};
```

DashboardBackend###0250-TicketStats

Parameters for the dashboard backend of the ticket stats of the agent interface. "Limit" is the number of entries shown by default. "Group" is used to restrict the access to the plugin (e. g. Group: admin;group1;group2;). "Default" determines if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTLLocal" is the cache time in minutes for the plugin.

Default value:

```
$Self->{'DashboardBackend'}->{'0250-TicketStats'} =  {
  'Block' => 'ContentSmall',
  'CacheTTLLocal' => '30',
  'Changed' => '1',
  'Closed' => '1',
  'Default' => '1',
  'Group' => '',
  'Module' => 'Kernel::Output::HTML::Dashboard::TicketStatsGeneric',
  'Permission' => 'rw',
  'Title' => '7 Day Stats'
};
```

DashboardBackend###0260-TicketCalendar

Parameters for the dashboard backend of the upcoming events widget of the agent interface. "Limit" is the number of entries shown by default. "Group" is used to restrict the access to the plugin (e. g. Group: admin;group1;group2;). "Default" determines if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTLLocal" is the cache time in minutes for the plugin.

Default value:

```
$Self->{'DashboardBackend'}->{'0260-TicketCalendar'} =  {
  'Block' => 'ContentSmall',
  'CacheTTL' => '2',
  'Default' => '1',
  'Group' => '',
  'Limit' => '6',
  'Module' => 'Kernel::Output::HTML::Dashboard::Calendar',
  'OwnerOnly' => '',
  'Permission' => 'rw',
  'Title' => 'Upcoming Events'
};
```

DashboardBackend###0270-TicketQueueOverview

Parameters for the dashboard backend of the queue overview widget of the agent interface. "Group" is used to restrict the access to the plugin (e. g. Group: admin;group1;group2;). "QueuePermissionGroup" is not mandatory, queues are only listed if they belong to this permission group if you enable it. "States" is a list of states, the key is the sort order of the state in the widget. "Default" determines if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTLLocal" is the cache time in minutes for the plugin.

Default value:

```
$Self->{'DashboardBackend'}->{'0270-TicketQueueOverview'} =  {
  'Block' => 'ContentLarge',
  'CacheTTLLocal' => '0.5',
  'Default' => '1',
  'Description' => 'Provides a matrix overview of the tickets per state per queue.',
  'Group' => '',
  'Module' => 'Kernel::Output::HTML::Dashboard::TicketQueueOverview',
  'Permission' => 'rw',
```

```
  'QueuePermissionGroup' => 'users',
  'Sort' => 'SortBy=Age;OrderBy=Up',
  'States' => {
    '1' => 'new',
    '4' => 'open',
    '6' => 'pending reminder'
  },
  'Title' => 'Ticket Queue Overview'
};
```

DashboardBackend###0280-DashboardEventsTicketCalendar

Parameters for the dashboard backend of the ticket events calendar of the agent interface. "Limit" is the number of entries shown by default. "Group" is used to restrict the access to the plugin (e. g. Group: admin;group1;group2;). "Default" determines if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTLLocal" is the cache time in minutes for the plugin.

Default value:

```
$Self->{'DashboardBackend'}->{'0280-DashboardEventsTicketCalendar'} =  {
  'Block' => 'ContentLarge',
  'CacheTTL' => '0',
  'Default' => '0',
  'Group' => '',
  'Module' => 'Kernel::Output::HTML::Dashboard::EventsTicketCalendar',
  'Title' => 'Events Ticket Calendar'
};
```

AgentCustomerInformationCenter::Backend###0100-CIC-TicketPendingReminder

Parameters for the dashboard backend of the ticket pending reminder overview of the agent interface . "Limit" is the number of entries shown by default. "Group" is used to restrict the access to the plugin (e. g. Group: admin;group1;group2;). "Default" determines if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTLLocal" is the cache time in minutes for the plugin. Note: Only Ticket attributes and Dynamic Fields (DynamicField_NameX) are allowed for DefaultColumns. Possible settings: 0 = Disabled, 1 = Available, 2 = Enabled by default.

Default value:

```
$Self->{'AgentCustomerInformationCenter::Backend'}->{'0100-CIC-TicketPendingReminder'}
=  {
  'Attributes' => 'TicketPendingTimeOlderMinutes=1;StateType=pending
reminder;SortBy=PendingTime;OrderBy=Down;',
  'Block' => 'ContentLarge',
  'CacheTTLLocal' => '0.5',
  'Default' => '1',
  'DefaultColumns' => {
    'Age' => '2',
    'Changed' => '1',
    'Created' => '1',
    'CustomerCompanyName' => '1',
    'CustomerID' => '1',
    'CustomerName' => '1',
    'CustomerUserID' => '1',
    'EscalationResponseTime' => '1',
    'EscalationSolutionTime' => '1',
    'EscalationTime' => '1',
    'EscalationUpdateTime' => '1',
    'Lock' => '1',
    'Owner' => '1',
    'PendingTime' => '1',
    'Priority' => '1',
    'Queue' => '1',
    'Responsible' => '1',
    'SLA' => '1',
    'Service' => '1',
    'State' => '1',
```

```
    'TicketNumber' => '2',
    'Title' => '2',
    'Type' => '1'
  },
  'Description' => 'All tickets with a reminder set where the reminder date has been
 reached',
  'Filter' => 'Locked',
  'Group' => '',
  'Limit' => '10',
  'Module' => 'Kernel::Output::HTML::Dashboard::TicketGeneric',
  'Permission' => 'ro',
  'Time' => 'UntilTime',
  'Title' => 'Reminder Tickets'
};
```

AgentCustomerInformationCenter::Backend###0110-CIC-TicketEscalation

Parameters for the dashboard backend of the ticket escalation overview of the agent interface . "Limit" is the number of entries shown by default. "Group" is used to restrict the access to the plugin (e. g. Group: admin;group1;group2;). "Default" determines if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTLLocal" is the cache time in minutes for the plugin. Note: Only Ticket attributes and Dynamic Fields (DynamicField_NameX) are allowed for DefaultColumns. Possible settings: 0 = Disabled, 1 = Available, 2 = Enabled by default.

Default value:

```
$Self->{'AgentCustomerInformationCenter::Backend'}->{'0110-CIC-TicketEscalation'} =  {
  'Attributes' =>
 'TicketEscalationTimeOlderMinutes=1;SortBy=EscalationTime;OrderBy=Down;',
  'Block' => 'ContentLarge',
  'CacheTTLLocal' => '0.5',
  'Default' => '1',
  'DefaultColumns' => {
    'Age' => '2',
    'Changed' => '1',
    'Created' => '1',
    'CustomerCompanyName' => '1',
    'CustomerID' => '1',
    'CustomerName' => '1',
    'CustomerUserID' => '1',
    'EscalationResponseTime' => '1',
    'EscalationSolutionTime' => '1',
    'EscalationTime' => '1',
    'EscalationUpdateTime' => '1',
    'Lock' => '1',
    'Owner' => '1',
    'PendingTime' => '1',
    'Priority' => '1',
    'Queue' => '1',
    'Responsible' => '1',
    'SLA' => '1',
    'Service' => '1',
    'State' => '1',
    'TicketNumber' => '2',
    'Title' => '2',
    'Type' => '1'
  },
  'Description' => 'All escalated tickets',
  'Filter' => 'All',
  'Group' => '',
  'Limit' => '10',
  'Module' => 'Kernel::Output::HTML::Dashboard::TicketGeneric',
  'Permission' => 'ro',
  'Time' => 'EscalationTime',
  'Title' => 'Escalated Tickets'
};
```

AgentCustomerInformationCenter::Backend###0120-CIC-TicketNew

Parameters for the dashboard backend of the new tickets overview of the agent interface. "Limit" is the number of entries shown by default. "Group" is used to restrict the

access to the plugin (e. g. Group: admin;group1;group2;). "Default" determines if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTLLocal" is the cache time in minutes for the plugin. Note: Only Ticket attributes and Dynamic Fields (DynamicField_NameX) are allowed for DefaultColumns. Possible settings: 0 = Disabled, 1 = Available, 2 = Enabled by default.

Default value:

```
$Self->{'AgentCustomerInformationCenter::Backend'}->{'0120-CIC-TicketNew'} =  {
 'Attributes' => 'StateType=new;',
 'Block' => 'ContentLarge',
 'CacheTTLLocal' => '0.5',
 'Default' => '1',
 'DefaultColumns' => {
   'Age' => '2',
   'Changed' => '1',
   'Created' => '1',
   'CustomerCompanyName' => '1',
   'CustomerID' => '1',
   'CustomerName' => '1',
   'CustomerUserID' => '1',
   'EscalationResponseTime' => '1',
   'EscalationSolutionTime' => '1',
   'EscalationTime' => '1',
   'EscalationUpdateTime' => '1',
   'Lock' => '1',
   'Owner' => '1',
   'PendingTime' => '1',
   'Priority' => '1',
   'Queue' => '1',
   'Responsible' => '1',
   'SLA' => '1',
   'Service' => '1',
   'State' => '1',
   'TicketNumber' => '2',
   'Title' => '2',
   'Type' => '1'
 },
 'Description' => 'All new tickets, these tickets have not been worked on yet',
 'Filter' => 'All',
 'Group' => '',
 'Limit' => '10',
 'Module' => 'Kernel::Output::HTML::Dashboard::TicketGeneric',
 'Permission' => 'ro',
 'Time' => 'Age',
 'Title' => 'New Tickets'
};
```

AgentCustomerInformationCenter::Backend###0130-CIC-TicketOpen

Parameters for the dashboard backend of the open tickets overview of the agent interface. "Limit" is the number of entries shown by default. "Group" is used to restrict the access to the plugin (e. g. Group: admin;group1;group2;). "Default" determines if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTLLocal" is the cache time in minutes for the plugin. Note: Only Ticket attributes and Dynamic Fields (DynamicField_NameX) are allowed for DefaultColumns. Possible settings: 0 = Disabled, 1 = Available, 2 = Enabled by default.

Default value:

```
$Self->{'AgentCustomerInformationCenter::Backend'}->{'0130-CIC-TicketOpen'} =  {
 'Attributes' => 'StateType=open;',
 'Block' => 'ContentLarge',
 'CacheTTLLocal' => '0.5',
 'Default' => '1',
 'DefaultColumns' => {
   'Age' => '2',
   'Changed' => '1',
   'Created' => '1',
   'CustomerCompanyName' => '1',
```

```
    'CustomerID' => '1',
    'CustomerName' => '1',
    'CustomerUserID' => '1',
    'EscalationResponseTime' => '1',
    'EscalationSolutionTime' => '1',
    'EscalationTime' => '1',
    'EscalationUpdateTime' => '1',
    'Lock' => '1',
    'Owner' => '1',
    'PendingTime' => '1',
    'Priority' => '1',
    'Queue' => '1',
    'Responsible' => '1',
    'SLA' => '1',
    'Service' => '1',
    'State' => '1',
    'TicketNumber' => '2',
    'Title' => '2',
    'Type' => '1'
  },
  'Description' => 'All open tickets, these tickets have already been worked on, but
 need a response',
  'Filter' => 'All',
  'Group' => '',
  'Limit' => '10',
  'Module' => 'Kernel::Output::HTML::Dashboard::TicketGeneric',
  'Permission' => 'ro',
  'Time' => 'Age',
  'Title' => 'Open Tickets / Need to be answered'
};
```

AgentCustomerInformationCenter::Backend###0500-CIC-CustomerIDStatus

Parameters for the dashboard backend of the customer id status widget of the agent interface . "Group" is used to restrict the access to the plugin (e. g. Group: admin;group1;group2;). "Default" determines if the plugin is enabled by default or if the user needs to enable it manually. "CacheTTLLocal" is the cache time in minutes for the plugin.

Default value:

```
$Self->{'AgentCustomerInformationCenter::Backend'}->{'0500-CIC-CustomerIDStatus'} =  {
  'Attributes' => '',
  'Block' => 'ContentSmall',
  'CacheTTLLocal' => '0.5',
  'Default' => '1',
  'Description' => 'Company Status',
  'Group' => '',
  'Module' => 'Kernel::Output::HTML::Dashboard::CustomerIDStatus',
  'Permission' => 'ro',
  'Title' => 'Company Status'
};
```

Ticket → Frontend::Agent::Dashboard::EventsTicketCalendar

DashboardEventsTicketCalendar###CalendarWidth

Defines the calendar width in percent. Default is 95%.

This setting can not be deactivated.

Default value:

```
$Self->{'DashboardEventsTicketCalendar'}->{'CalendarWidth'} =  '95';
```

DashboardEventsTicketCalendar###Queues

Defines queues that's tickets are used for displaying as calendar events.

This setting can not be deactivated.

Default value:

```
$Self->{'DashboardEventsTicketCalendar'}->{'Queues'} =  [
  'Raw'
];
```

DashboardEventsTicketCalendar::DynamicFieldStartTime

Define dynamic field name for start time. This field has to be manually added to the system as Ticket: "Date / Time" and must be activated in ticket creation screens and/or in any other ticket action screens.

Default value:

```
$Self->{'DashboardEventsTicketCalendar::DynamicFieldStartTime'} =
'TicketCalendarStartTime';
```

DashboardEventsTicketCalendar::DynamicFieldEndTime

Define dynamic field name for end time. This field has to be manually added to the system as Ticket: "Date / Time" and must be activated in ticket creation screens and/or in any other ticket action screens.

Default value:

```
$Self->{'DashboardEventsTicketCalendar::DynamicFieldEndTime'} =
'TicketCalendarEndTime';
```

DashboardEventsTicketCalendar::DynamicFieldsForEvents

Defines the dynamic fields that are used for displaying on calendar events.

This setting can not be deactivated.

Default value:

```
$Self->{'DashboardEventsTicketCalendar::DynamicFieldsForEvents'} =  [
  'TicketCalendarStartTime',
  'TicketCalendarEndTime'
];
```

DashboardEventsTicketCalendar::TicketFieldsForEvents

Defines the ticket fields that are going to be displayed calendar events. The "Key" defines the field or ticket attribute and the "Content" defines the display name.

This setting can not be deactivated.

Default value:

```
$Self->{'DashboardEventsTicketCalendar::TicketFieldsForEvents'} =  {
  'CustomerID' => 'Customer ID',
  'CustomerUserID' => 'Customer user',
  'Priority' => 'Priority',
  'Queue' => 'Queue',
  'SLA' => 'SLA',
  'Service' => 'Service',
  'State' => 'State',
  'Title' => 'Title',
  'Type' => 'Type'
};
```

Ticket → Frontend::Agent::Dashboard::TicketFilters

OnlyValuesOnTicket

Defines if the values for filters should be retrieved from all available tickets. If set to "Yes", only values which are actually used in any ticket will be available for filtering. Please note: The list of customers will always be retrieved like this.

This setting can not be deactivated.

Default value:

```
$Self->{'OnlyValuesOnTicket'} =  '1';
```

Ticket → Frontend::Agent::LinkObject

LinkObject::ComplexTable::SettingsVisibility###Ticket

Define Actions where a settings button is available in the linked objects widget (LinkObject::ViewMode = "complex"). Please note that these Actions must have registered the following JS and CSS files: Core.AllocationList.css, Core.UI.AllocationList.js, Core.UI.Table.Sort.js, Core.Agent.TableFilters.js.

Default value:

```
$Self->{'LinkObject::ComplexTable::SettingsVisibility'}->{'Ticket'} =  [
  'AgentTicketZoom'
];
```

LinkObject::ComplexTable###Ticket

Define which columns are shown in the linked tickets widget (LinkObject::ViewMode = "complex"). Note: Only Ticket attributes and Dynamic Fields (DynamicField_NameX) are allowed for DefaultColumns. Possible settings: 0 = Disabled, 1 = Available, 2 = Enabled by default.

Default value:

```
$Self->{'LinkObject::ComplexTable'}->{'Ticket'} =  {
  'DefaultColumns' => {
    'Age' => '1',
    'Changed' => '1',
    'Created' => '2',
    'CustomerID' => '1',
    'CustomerName' => '1',
    'CustomerUserID' => '1',
    'EscalationResponseTime' => '1',
    'EscalationSolutionTime' => '1',
    'EscalationTime' => '1',
    'EscalationUpdateTime' => '1',
    'Lock' => '1',
    'Owner' => '1',
    'PendingTime' => '1',
    'Priority' => '1',
    'Queue' => '2',
    'Responsible' => '1',
    'SLA' => '1',
    'Service' => '1',
    'State' => '2',
    'TicketNumber' => '2',
    'Title' => '2',
    'Type' => '1'
  },
  'Module' => 'Kernel::Output::HTML::LinkObject::Ticket.pm',
  'Priority' => {
    'Age' => '110',
    'Changed' => '120',
    'Created' => '310',
    'CustomerID' => '240',
    'CustomerName' => '250',
    'CustomerUserID' => '260',
    'EscalationResponseTime' => '160',
    'EscalationSolutionTime' => '150',
    'EscalationTime' => '140',
    'EscalationUpdateTime' => '170',
    'Lock' => '200',
    'Owner' => '220',
    'PendingTime' => '130',
    'Priority' => '300',
    'Queue' => '210',
```

```
    'Responsible' => '230',
    'SLA' => '290',
    'Service' => '280',
    'State' => '190',
    'TicketNumber' => '100',
    'Title' => '180',
    'Type' => '270'
  }
};
```

Ticket → Frontend::Agent::ModuleMetaHead

Frontend::HeaderMetaModule###2-TicketSearch

Module to generate html OpenSearch profile for short ticket search in the agent interface.

Default value:

```
$Self->{'Frontend::HeaderMetaModule'}->{'2-TicketSearch'} =  {
  'Action' => 'AgentTicketSearch',
  'Module' => 'Kernel::Output::HTML::HeaderMeta::AgentTicketSearch'
};
```

Ticket → Frontend::Agent::ModuleNotify

Frontend::NotifyModule###5-Ticket::TicketEscalation

Module to show notifications and escalations (ShownMax: max. shown escalations, EscalationInMinutes: Show ticket which will escalation in, CacheTime: Cache of calculated escalations in seconds).

This setting is not active by default.

Default value:

```
$Self->{'Frontend::NotifyModule'}->{'5-Ticket::TicketEscalation'} =  {
  'CacheTime' => '40',
  'EscalationInMinutes' => '120',
  'Module' => 'Kernel::Output::HTML::Notification::AgentTicketEscalation',
  'ShownMax' => '25'
};
```

Ticket → Frontend::Agent::ModuleRegistration

Frontend::Module###AgentTicketQueue

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketQueue'} =  {
  'Description' => 'Overview of all open Tickets.',
  'Loader' => {
    'CSS' => [
      'Core.AgentTicketQueue.css',
      'Core.AllocationList.css'
    ],
    'JavaScript' => [
      'Core.UI.AllocationList.js',
      'Core.Agent.TableFilters.js'
    ]
  },
  'NavBar' => [
    {
      'AccessKey' => 'o',
      'Block' => '',
      'Description' => 'Overview of all open Tickets.',
      'Link' => 'Action=AgentTicketQueue',
```

```
      'LinkOption' => '',
      'Name' => 'Queue view',
      'NavBar' => 'Ticket',
      'Prio' => '100',
      'Type' => ''
    },
    {
      'AccessKey' => 't',
      'Block' => 'ItemArea',
      'Description' => '',
      'Link' => 'Action=AgentTicketQueue',
      'LinkOption' => '',
      'Name' => 'Tickets',
      'NavBar' => 'Ticket',
      'Prio' => '200',
      'Type' => 'Menu'
    }
  ],
  'NavBarName' => 'Ticket',
  'Title' => 'QueueView'
};
```

Frontend::Module###AgentTicketService

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketService'} =  {
  'Description' => 'Overview of all open Tickets.',
  'Loader' => {
    'CSS' => [
      'Core.AgentTicketService.css',
      'Core.AllocationList.css'
    ],
    'JavaScript' => [
      'Core.UI.AllocationList.js',
      'Core.Agent.TableFilters.js'
    ]
  },
  'NavBar' => [
    {
      'AccessKey' => 'O',
      'Block' => '',
      'Description' => 'Overview of all open Tickets.',
      'Link' => 'Action=AgentTicketService',
      'LinkOption' => '',
      'Name' => 'Service view',
      'NavBar' => 'Ticket',
      'Prio' => '105',
      'Type' => ''
    }
  ],
  'NavBarName' => 'Ticket',
  'Title' => 'ServiceView'
};
```

Frontend::Module###AgentTicketPhone

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketPhone'} =  {
  'Description' => 'Create new phone ticket.',
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.CustomerSearch.js',
      'Core.Agent.TicketAction.js'
    ]
  },
  'NavBar' => [
    {
```

```
      'AccessKey' => 'n',
      'Block' => '',
      'Description' => 'Create new phone ticket (inbound).',
      'Link' => 'Action=AgentTicketPhone',
      'LinkOption' => '',
      'Name' => 'New phone ticket',
      'NavBar' => 'Ticket',
      'Prio' => '200',
      'Type' => ''
    }
  ],
  'NavBarName' => 'Ticket',
  'Title' => 'New phone ticket'
};
```

Frontend::Module###AgentTicketPhoneOutbound

Frontend module registration for the agent interface.

Default value:

```
 $Self->{'Frontend::Module'}->{'AgentTicketPhoneOutbound'} =  {
  'Description' => 'Phone Call.',
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.TicketAction.js'
    ]
  },
  'NavBarName' => 'Ticket',
  'Title' => 'Phone-Ticket'
};
```

Frontend::Module###AgentTicketPhoneInbound

Frontend module registration for the agent interface.

Default value:

```
 $Self->{'Frontend::Module'}->{'AgentTicketPhoneInbound'} =  {
  'Description' => 'Incoming Phone Call.',
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.TicketAction.js'
    ]
  },
  'NavBarName' => 'Ticket',
  'Title' => 'Phone-Ticket'
};
```

Frontend::Module###AgentTicketEmail

Frontend module registration for the agent interface.

Default value:

```
 $Self->{'Frontend::Module'}->{'AgentTicketEmail'} =  {
  'Description' => 'Create new email ticket.',
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.CustomerSearch.js',
      'Core.Agent.TicketAction.js'
    ]
  },
  'NavBar' => [
    {
      'AccessKey' => 'm',
      'Block' => '',
      'Description' => 'Create new email ticket and send this out (outbound).',
      'Link' => 'Action=AgentTicketEmail',
      'LinkOption' => '',
      'Name' => 'New email ticket',
      'NavBar' => 'Ticket',
      'Prio' => '210',
```

```
      'Type' => ''
    }
  ],
  'NavBarName' => 'Ticket',
  'Title' => 'New email ticket'
};
```

Frontend::Module###AgentTicketSearch

Frontend module registration for the agent interface.

Default value:

```
 $Self->{'Frontend::Module'}->{'AgentTicketSearch'} =  {
  'Description' => 'Search Ticket.',
  'Loader' => {
    'JavaScript' => [
      'Core.UI.AllocationList.js',
      'Core.Agent.TableFilters.js'
    ]
  },
  'NavBar' => [
    {
      'AccessKey' => 's',
      'Block' => '',
      'Description' => 'Search Tickets.',
      'Link' => 'Action=AgentTicketSearch',
      'LinkOption' => 'onclick="window.setTimeout(function()
{Core.Agent.Search.OpenSearchDialog(\'AgentTicketSearch\');}, 0); return false;"',
      'Name' => 'Search',
      'NavBar' => 'Ticket',
      'Prio' => '300',
      'Type' => ''
    }
  ],
  'NavBarName' => 'Ticket',
  'Title' => 'Search'
};
```

Frontend::Module###AgentTicketLockedView

Frontend module registration for the agent interface.

Default value:

```
 $Self->{'Frontend::Module'}->{'AgentTicketLockedView'} =  {
  'Description' => 'Locked Tickets.',
  'Loader' => {
    'CSS' => [
      'Core.AgentTicketQueue.css',
      'Core.AllocationList.css'
    ],
    'JavaScript' => [
      'Core.UI.AllocationList.js',
      'Core.Agent.TableFilters.js'
    ]
  },
  'NavBarName' => 'Ticket',
  'Title' => 'Locked Tickets'
};
```

Frontend::Module###AgentTicketResponsibleView

Frontend module registration for the agent interface.

Default value:

```
 $Self->{'Frontend::Module'}->{'AgentTicketResponsibleView'} =  {
  'Description' => 'Responsible Tickets.',
  'Loader' => {
    'CSS' => [
      'Core.AllocationList.css'
    ],
```

```
    'JavaScript' => [
      'Core.UI.AllocationList.js',
      'Core.Agent.TableFilters.js'
    ]
  },
  'NavBarName' => 'Ticket',
  'Title' => 'Responsible Tickets'
};
```

Frontend::Module###AgentTicketWatchView
Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketWatchView'} =  {
  'Description' => 'Watched Tickets.',
  'Loader' => {
    'CSS' => [
      'Core.AgentTicketQueue.css',
      'Core.AllocationList.css'
    ],
    'JavaScript' => [
      'Core.UI.AllocationList.js',
      'Core.Agent.TableFilters.js'
    ]
  },
  'NavBarName' => 'Ticket',
  'Title' => 'Watched Tickets'
};
```

Frontend::Module###AgentCustomerSearch
Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentCustomerSearch'} =  {
  'Description' => 'AgentCustomerSearch.',
  'NavBarName' => 'Ticket',
  'Title' => 'AgentCustomerSearch'
};
```

Frontend::Module###AgentUserSearch
Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentUserSearch'} =  {
  'Description' => 'AgentUserSearch.',
  'NavBarName' => 'Ticket',
  'Title' => 'AgentUserSearch'
};
```

Frontend::Module###AgentTicketStatusView
Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketStatusView'} =  {
  'Description' => 'Overview of all open tickets.',
  'Loader' => {
    'CSS' => [
      'Core.AllocationList.css'
    ],
    'JavaScript' => [
      'Core.UI.AllocationList.js',
      'Core.Agent.TableFilters.js'
    ]
  },
  'NavBar' => [
```

```
    {
      'AccessKey' => 'v',
      'Block' => '',
      'Description' => 'Overview of all open Tickets.',
      'Link' => 'Action=AgentTicketStatusView',
      'LinkOption' => '',
      'Name' => 'Status view',
      'NavBar' => 'Ticket',
      'Prio' => '110',
      'Type' => ''
    }
  ],
  'NavBarName' => 'Ticket',
  'Title' => 'Status view'
};
```

Frontend::Module###AgentTicketEscalationView

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketEscalationView'} =  {
  'Description' => 'Overview of all escalated tickets.',
  'Loader' => {
    'CSS' => [
      'Core.AllocationList.css'
    ],
    'JavaScript' => [
      'Core.UI.AllocationList.js',
      'Core.Agent.TableFilters.js'
    ]
  },
  'NavBar' => [
    {
      'AccessKey' => 'e',
      'Block' => '',
      'Description' => 'Overview Escalated Tickets.',
      'Link' => 'Action=AgentTicketEscalationView',
      'LinkOption' => '',
      'Name' => 'Escalation view',
      'NavBar' => 'Ticket',
      'Prio' => '120',
      'Type' => ''
    }
  ],
  'NavBarName' => 'Ticket',
  'Title' => 'Escalation view'
};
```

Frontend::Module###AgentZoom

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentZoom'} =  {
  'Description' => 'Compat module for AgentZoom to AgentTicketZoom.',
  'NavBarName' => 'Ticket',
  'Title' => ''
};
```

Frontend::Module###AgentTicketZoom

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketZoom'} =  {
  'Description' => 'Ticket Zoom.',
  'Loader' => {
    'CSS' => [
      'Core.Agent.TicketProcess.css',
```

```
      'Core.Agent.TicketMenuModuleCluster.css',
      'Core.AllocationList.css'
    ],
    'JavaScript' => [
      'thirdparty/jquery-tablesorter-2.0.5/jquery.tablesorter.js',
      'Core.Agent.TicketZoom.js',
      'Core.UI.AllocationList.js',
      'Core.UI.Table.Sort.js',
      'Core.Agent.TableFilters.js',
      'Core.Agent.LinkObject.js'
    ]
  },
  'NavBarName' => 'Ticket',
  'Title' => 'Zoom'
};
```

Frontend::Module###AgentTicketAttachment

Frontend module registration for the agent interface.

Default value:

```
 $Self->{'Frontend::Module'}->{'AgentTicketAttachment'} =  {
  'Description' => 'To download attachments.',
  'NavBarName' => 'Ticket',
  'Title' => ''
};
```

Frontend::Module###AgentTicketPlain

Frontend module registration for the agent interface.

Default value:

```
 $Self->{'Frontend::Module'}->{'AgentTicketPlain'} =  {
  'Description' => 'Ticket plain view of an email.',
  'NavBarName' => 'Ticket',
  'Title' => 'Plain'
};
```

Frontend::Module###AgentTicketNote

Frontend module registration for the agent interface.

Default value:

```
 $Self->{'Frontend::Module'}->{'AgentTicketNote'} =  {
  'Description' => 'Ticket Note.',
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.TicketAction.js'
    ]
  },
  'NavBarName' => 'Ticket',
  'Title' => 'Note'
};
```

Frontend::Module###AgentTicketMerge

Frontend module registration for the agent interface.

Default value:

```
 $Self->{'Frontend::Module'}->{'AgentTicketMerge'} =  {
  'Description' => 'Ticket Merge.',
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.TicketMerge.js'
    ]
  },
  'NavBarName' => 'Ticket',
  'Title' => 'Merge'
};
```

Frontend::Module###AgentTicketPending

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketPending'} =  {
  'Description' => 'Ticket Pending.',
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.TicketAction.js'
    ]
  },
  'NavBarName' => 'Ticket',
  'Title' => 'Pending'
};
```

Frontend::Module###AgentTicketWatcher

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketWatcher'} =  {
  'Description' => 'A TicketWatcher Module.',
  'NavBarName' => 'Ticket-Watcher',
  'Title' => 'Ticket Watcher'
};
```

Frontend::Module###AgentTicketPriority

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketPriority'} =  {
  'Description' => 'Ticket Priority.',
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.TicketAction.js'
    ]
  },
  'NavBarName' => 'Ticket',
  'Title' => 'Priority'
};
```

Frontend::Module###AgentTicketLock

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketLock'} =  {
  'Description' => 'Ticket Lock.',
  'NavBarName' => 'Ticket',
  'Title' => 'Lock'
};
```

Frontend::Module###AgentTicketMove

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketMove'} =  {
  'Description' => 'Ticket Move.',
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.TicketAction.js'
    ]
  },
  'NavBarName' => 'Ticket',
  'Title' => 'Move'
};
```

Frontend::Module###AgentTicketHistory

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketHistory'} =  {
  'Description' => 'Ticket History.',
  'NavBarName' => 'Ticket',
  'Title' => 'History'
};
```

Frontend::Module###AgentTicketOwner

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketOwner'} =  {
  'Description' => 'Ticket Owner.',
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.TicketAction.js'
    ]
  },
  'NavBarName' => 'Ticket',
  'Title' => 'Owner'
};
```

Frontend::Module###AgentTicketResponsible

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketResponsible'} =  {
  'Description' => 'Ticket Responsible.',
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.TicketAction.js'
    ]
  },
  'NavBarName' => 'Ticket',
  'Title' => 'Responsible'
};
```

Frontend::Module###AgentTicketCompose

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketCompose'} =  {
  'Description' => 'Ticket Compose email Answer.',
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.CustomerSearch.js',
      'Core.Agent.TicketAction.js'
    ]
  },
  'NavBarName' => 'Ticket',
  'Title' => 'Compose'
};
```

Frontend::Module###AgentTicketBounce

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketBounce'} =  {
  'Description' => 'Ticket Compose Bounce Email.',
  'NavBarName' => 'Ticket',
  'Title' => 'Bounce'
};
```

Frontend::Module###AgentTicketForward

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketForward'} =  {
  'Description' => 'Ticket Forward Email.',
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.CustomerSearch.js',
      'Core.Agent.TicketAction.js'
    ]
  },
  'NavBarName' => 'Ticket',
  'Title' => 'Forward'
};
```

Frontend::Module###AgentTicketEmailOutbound

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketEmailOutbound'} =  {
  'Description' => 'Ticket Outbound Email.',
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.CustomerSearch.js',
      'Core.Agent.TicketAction.js'
    ]
  },
  'NavBarName' => 'Ticket',
  'Title' => 'Email Outbound'
};
```

Frontend::Module###AgentTicketCustomer

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketCustomer'} =  {
  'Description' => 'Ticket Customer.',
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.CustomerSearch.js',
      'Core.Agent.TicketAction.js'
    ]
  },
  'NavBarName' => 'Ticket',
  'Title' => 'Customer'
};
```

Frontend::Module###AgentTicketClose

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketClose'} =  {
  'Description' => 'Ticket Close.',
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.TicketAction.js'
    ]
  },
  'NavBarName' => 'Ticket',
  'Title' => 'Close'
};
```

Frontend::Module###AgentTicketFreeText

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketFreeText'} =  {
  'Description' => 'Ticket FreeText.',
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.TicketAction.js'
    ]
  },
  'NavBarName' => 'Ticket',
  'Title' => 'Free Fields'
};
```

Frontend::Module###AgentTicketPrint

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketPrint'} =  {
  'Description' => 'Ticket Print.',
  'NavBarName' => 'Ticket',
  'Title' => 'Print'
};
```

Frontend::Module###AgentTicketBulk

Frontend module registration for the agent interface.

Default value:

```
$Self->{'Frontend::Module'}->{'AgentTicketBulk'} =  {
  'Description' => 'Ticket bulk module.',
  'Loader' => {
    'JavaScript' => [
      'Core.Agent.TicketAction.js'
    ]
  },
  'NavBarName' => 'Ticket',
  'Title' => 'Bulk Action'
};
```

Ticket → Frontend::Agent::Preferences

PreferencesGroups###CustomQueue

Parameters for the CustomQueue object in the preference view of the agent interface.

Default value:

```
$Self->{'PreferencesGroups'}->{'CustomQueue'} =  {
  'Active' => '1',
  'Column' => 'Notification Settings',
  'Desc' => 'Your queue selection of your preferred queues. You also get notified about
 those queues via email if enabled.',
  'Key' => '',
  'Label' => 'My Queues',
  'Module' => 'Kernel::Output::HTML::Preferences::CustomQueue',
  'Permission' => 'ro',
  'Prio' => '1000'
};
```

PreferencesGroups###CustomService

Parameters for the CustomService object in the preference view of the agent interface.

Default value:

```
$Self->{'PreferencesGroups'}->{'CustomService'} =  {
  'Active' => '1',
  'Column' => 'Notification Settings',
  'Desc' => 'Your service selection of your preferred services. You also get notified
 about those services via email if enabled.',
```

```
  'Key' => '',
  'Label' => 'My Services',
  'Module' => 'Kernel::Output::HTML::Preferences::CustomService',
  'Prio' => '1000'
};
```

PreferencesGroups###RefreshTime

Parameters for the RefreshTime object in the preference view of the agent interface.

Default value:

```
$Self->{'PreferencesGroups'}->{'RefreshTime'} =  {
  'Active' => '1',
  'Column' => 'Other Settings',
  'Data' => {
    '0' => 'off',
    '10' => '10 minutes',
    '15' => '15 minutes',
    '2' => ' 2 minutes',
    '5' => ' 5 minutes',
    '7' => ' 7 minutes'
  },
  'DataSelected' => '0',
  'Desc' => 'If enabled, the different overviews (Dashboard, LockedView, QueueView) will
automatically refresh after the specified time.',
  'Key' => 'After',
  'Label' => 'Overview Refresh Time',
  'Module' => 'Kernel::Output::HTML::Preferences::Generic',
  'PrefKey' => 'UserRefreshTime',
  'Prio' => '2000'
};
```

PreferencesGroups###TicketOverviewSmallPageShown

Parameters for the pages (in which the tickets are shown) of the small ticket overview.

Default value:

```
$Self->{'PreferencesGroups'}->{'TicketOverviewSmallPageShown'} =  {
  'Active' => '0',
  'Column' => 'Other Settings',
  'Data' => {
    '10' => '10',
    '15' => '15',
    '20' => '20',
    '25' => '25',
    '30' => '30',
    '35' => '35'
  },
  'DataSelected' => '25',
  'Key' => 'Ticket limit per page for Ticket Overview "Small"',
  'Label' => 'Ticket Overview "Small" Limit',
  'Module' => 'Kernel::Output::HTML::Preferences::Generic',
  'PrefKey' => 'UserTicketOverviewSmallPageShown',
  'Prio' => '8000'
};
```

PreferencesGroups###TicketOverviewFilterSettings

Parameters for the column filters of the small ticket overview.

Default value:

```
$Self->{'PreferencesGroups'}->{'TicketOverviewFilterSettings'} =  {
  'Active' => '0',
  'Column' => 'Other Settings',
  'Key' => 'Column ticket filters for Ticket Overviews type "Small".',
  'Label' => 'Enabled filters.',
  'Module' => 'Kernel::Output::HTML::Preferences::ColumnFilters',
  'PrefKey' => 'UserFilterColumnsEnabled',
  'Prio' => '8100'
};
```

PreferencesGroups###TicketOverviewMediumPageShown

Parameters for the pages (in which the tickets are shown) of the medium ticket overview.

Default value:

```
$Self->{'PreferencesGroups'}->{'TicketOverviewMediumPageShown'} =  {
  'Active' => '0',
  'Column' => 'Other Settings',
  'Data' => {
    '10' => '10',
    '15' => '15',
    '20' => '20',
    '25' => '25',
    '30' => '30',
    '35' => '35'
  },
  'DataSelected' => '20',
  'Key' => 'Ticket limit per page for Ticket Overview "Medium"',
  'Label' => 'Ticket Overview "Medium" Limit',
  'Module' => 'Kernel::Output::HTML::Preferences::Generic',
  'PrefKey' => 'UserTicketOverviewMediumPageShown',
  'Prio' => '8100'
};
```

PreferencesGroups###TicketOverviewPreviewPageShown

Parameters for the pages (in which the tickets are shown) of the ticket preview overview.

Default value:

```
$Self->{'PreferencesGroups'}->{'TicketOverviewPreviewPageShown'} =  {
  'Active' => '0',
  'Column' => 'Other Settings',
  'Data' => {
    '10' => '10',
    '15' => '15',
    '20' => '20',
    '25' => '25',
    '30' => '30',
    '35' => '35'
  },
  'DataSelected' => '15',
  'Key' => 'Ticket limit per page for Ticket Overview "Preview"',
  'Label' => 'Ticket Overview "Preview" Limit',
  'Module' => 'Kernel::Output::HTML::Preferences::Generic',
  'PrefKey' => 'UserTicketOverviewPreviewPageShown',
  'Prio' => '8200'
};
```

PreferencesGroups###CreateNextMask

Parameters for the CreateNextMask object in the preference view of the agent interface.

Default value:

```
$Self->{'PreferencesGroups'}->{'CreateNextMask'} =  {
  'Active' => '1',
  'Column' => 'Other Settings',
  'Data' => {
    '0' => 'CreateTicket',
    'AgentTicketZoom' => 'TicketZoom'
  },
  'DataSelected' => '',
  'Desc' => 'Configure which screen should be shown after a new ticket has been
created.',
  'Key' => 'Screen',
  'Label' => 'Screen after new ticket',
  'Module' => 'Kernel::Output::HTML::Preferences::Generic',
  'PrefKey' => 'UserCreateNextMask',
```

```
  'Prio' => '3000'
};
```

PreferencesGroups###NotificationEvent
Transport selection for ticket notifications.

Default value:

```
$Self->{'PreferencesGroups'}->{'NotificationEvent'} =  {
  'Active' => '1',
  'Column' => 'Notification Settings',
  'Desc' => 'Choose for which kind of ticket changes you want to receive
 notifications.',
  'Label' => 'Ticket notifications',
  'Module' => 'Kernel::Output::HTML::Preferences::NotificationEvent',
  'PrefKey' => 'AdminNotifcationEventTransport',
  'Prio' => '8000'
};
```

Ticket → Frontend::Agent::SearchRouter

Frontend::Search###AgentCustomerInformationCenter
Search backend router.

Default value:

```
$Self->{'Frontend::Search'}->{'AgentCustomerInformationCenter'} =  {
  '^AgentCustomerInformationCenter' => 'Action=AgentCustomerInformationCenterSearch'
};
```

Frontend::Search::JavaScript###AgentCustomerInformationCenter
JavaScript function for the search frontend.

Default value:

```
$Self->{'Frontend::Search::JavaScript'}->{'AgentCustomerInformationCenter'} =  {
  '^AgentCustomerInformationCenter' =>
 'Core.Agent.CustomerInformationCenterSearch.OpenSearchDialog()'
};
```

Frontend::Search###Ticket
Search backend router.

Default value:

```
$Self->{'Frontend::Search'}->{'Ticket'} =  {
  '^AgentTicket' => 'Action=AgentTicketSearch;Subaction=AJAX'
};
```

Ticket → Frontend::Agent::Ticket::ArticleAttachmentModule

Ticket::Frontend::ArticleAttachmentModule###1-Download
Shows a link to download article attachments in the zoom view of the article in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::ArticleAttachmentModule'}->{'1-Download'} =  {
  'Module' => 'Kernel::Output::HTML::ArticleAttachment::Download'
};
```

Ticket::Frontend::ArticleAttachmentModule###2-HTML-Viewer
Shows a link to access article attachments via a html online viewer in the zoom view of the article in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::ArticleAttachmentModule'}->{'2-HTML-Viewer'} =  {
  'Module' => 'Kernel::Output::HTML::ArticleAttachment::HTMLViewer'
};
```

Ticket → Frontend::Agent::Ticket::ArticleComposeModule

Ticket::Frontend::ArticleComposeModule###1-SignEmail
Module to compose signed messages (PGP or S/MIME).

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::ArticleComposeModule'}->{'1-SignEmail'} =  {
  'Module' => 'Kernel::Output::HTML::ArticleCompose::Sign'
};
```

Ticket::Frontend::ArticleComposeModule###2-CryptEmail
Module to crypt composed messages (PGP or S/MIME).

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::ArticleComposeModule'}->{'2-CryptEmail'} =  {
  'Module' => 'Kernel::Output::HTML::ArticleCompose::Crypt'
};
```

Ticket → Frontend::Agent::Ticket::ArticleViewModule

Ticket::Frontend::ArticleViewModule###1-PGP
Agent interface article notification module to check PGP.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::ArticleViewModule'}->{'1-PGP'} =  {
  'Module' => 'Kernel::Output::HTML::ArticleCheck::PGP'
};
```

Ticket::Frontend::ArticleViewModule###1-SMIME
Agent interface module to check incoming emails in the Ticket-Zoom-View if the S/MIME-key is available and true.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::ArticleViewModule'}->{'1-SMIME'} =  {
  'Module' => 'Kernel::Output::HTML::ArticleCheck::SMIME'
};
```

Ticket → Frontend::Agent::Ticket::ArticleViewModulePre

Ticket::Frontend::ArticlePreViewModule###1-PGP
Agent interface article notification module to check PGP.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::ArticlePreViewModule'}->{'1-PGP'} =  {
  'Module' => 'Kernel::Output::HTML::ArticleCheck::PGP'
};
```

Ticket::Frontend::ArticlePreViewModule###1-SMIME

Agent interface article notification module to check S/MIME.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::ArticlePreViewModule'}->{'1-SMIME'} =  {
  'Module' => 'Kernel::Output::HTML::ArticleCheck::SMIME'
};
```

Ticket → Frontend::Agent::Ticket::MenuModule

Ticket::Frontend::MenuModule###000-Back

Shows a link in the menu to go back in the ticket zoom view of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2". To cluster menu items use for Key "ClusterName" and for the Content any name you want to see in the UI. Use "ClusterPriority" to configure the order of a certain cluster within the toolbar.

Default value:

```
$Self->{'Ticket::Frontend::MenuModule'}->{'000-Back'} =  {
  'Action' => '',
  'ClusterName' => '',
  'ClusterPriority' => '',
  'Description' => 'Go back',
  'Link' => '[% Env("LastScreenOverview") %];TicketID=[% Data.TicketID | html %]',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'Back',
  'PopupType' => '',
  'Target' => ''
};
```

Ticket::Frontend::MenuModule###100-Lock

Shows a link in the menu to lock/unlock tickets in the ticket zoom view of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2". To cluster menu items use for Key "ClusterName" and for the Content any name you want to see in the UI. Use "ClusterPriority" to configure the order of a certain cluster within the toolbar.

Default value:

```
$Self->{'Ticket::Frontend::MenuModule'}->{'100-Lock'} =  {
  'Action' => 'AgentTicketLock',
  'ClusterName' => 'Miscellaneous',
  'ClusterPriority' => '800',
  'Description' => 'Lock / unlock this ticket',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Lock',
  'Name' => 'Lock',
  'Target' => ''
};
```

Ticket::Frontend::MenuModule###200-History

Shows a link in the menu to access the history of a ticket in the ticket zoom view of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2". To cluster menu items use for Key "ClusterName" and for the Content any name you want to see in the UI. Use "ClusterPriority" to configure the order of a certain cluster within the toolbar.

Default value:

```
$Self->{'Ticket::Frontend::MenuModule'}->{'200-History'} =  {
  'Action' => 'AgentTicketHistory',
  'ClusterName' => 'Miscellaneous',
  'ClusterPriority' => '800',
  'Description' => 'Show the history for this ticket',
  'Link' => 'Action=AgentTicketHistory;TicketID=[% Data.TicketID | html %]',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'History',
  'PopupType' => 'TicketHistory',
  'Target' => ''
};
```

Ticket::Frontend::MenuModule###210-Print

Shows a link in the menu to print a ticket or an article in the ticket zoom view of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2". To cluster menu items use for Key "ClusterName" and for the Content any name you want to see in the UI. Use "ClusterPriority" to configure the order of a certain cluster within the toolbar.

Default value:

```
$Self->{'Ticket::Frontend::MenuModule'}->{'210-Print'} =  {
  'Action' => 'AgentTicketPrint',
  'ClusterName' => '',
  'ClusterPriority' => '',
  'Description' => 'Print this ticket',
  'Link' => 'Action=AgentTicketPrint;TicketID=[% Data.TicketID | html %]',
  'LinkParam' => 'target="print"',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'Print',
  'PopupType' => 'TicketAction',
  'Target' => ''
};
```

Ticket::Frontend::MenuModule###300-Priority

Shows a link in the menu to see the priority of a ticket in the ticket zoom view of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2". To cluster menu items use for Key "ClusterName" and for the Content any name you want to see in the UI. Use "ClusterPriority" to configure the order of a certain cluster within the toolbar.

Default value:

```
$Self->{'Ticket::Frontend::MenuModule'}->{'300-Priority'} =  {
  'Action' => 'AgentTicketPriority',
  'ClusterName' => '',
  'ClusterPriority' => '',
  'Description' => 'Change the priority for this ticket',
  'Link' => 'Action=AgentTicketPriority;TicketID=[% Data.TicketID | html %]',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'Priority',
  'PopupType' => 'TicketAction',
  'Target' => ''
};
```

Ticket::Frontend::MenuModule###310-FreeText

Shows a link in the menu to add a free text field in the ticket zoom view of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2". To cluster menu items use for Key "ClusterName" and for the Content any name you want to see in the UI. Use "ClusterPriority" to configure the order of a certain cluster within the toolbar.

Default value:

```
$Self->{'Ticket::Frontend::MenuModule'}->{'310-FreeText'} =  {
  'Action' => 'AgentTicketFreeText',
  'ClusterName' => 'Miscellaneous',
```

```
  'ClusterPriority' => '800',
  'Description' => 'Change the free fields for this ticket',
  'Link' => 'Action=AgentTicketFreeText;TicketID=[% Data.TicketID | html %]',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'Free Fields',
  'PopupType' => 'TicketAction',
  'Target' => ''
};
```

Ticket::Frontend::MenuModule###320-Link

Shows a link in the menu that allows linking a ticket with another object in the ticket zoom view of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2". To cluster menu items use for Key "ClusterName" and for the Content any name you want to see in the UI. Use "ClusterPriority" to configure the order of a certain cluster within the toolbar.

Default value:

```
$Self->{'Ticket::Frontend::MenuModule'}->{'320-Link'} =  {
  'Action' => 'AgentLinkObject',
  'ClusterName' => 'Miscellaneous',
  'ClusterPriority' => '800',
  'Description' => 'Link this ticket to other objects',
  'Link' => 'Action=AgentLinkObject;SourceObject=Ticket;SourceKey=[% Data.TicketID |
 html %]',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'Link',
  'PopupType' => 'TicketAction',
  'Target' => ''
};
```

Ticket::Frontend::MenuModule###400-Owner

Shows a link in the menu to see the owner of a ticket in the ticket zoom view of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2". To cluster menu items use for Key "ClusterName" and for the Content any name you want to see in the UI. Use "ClusterPriority" to configure the order of a certain cluster within the toolbar.

Default value:

```
$Self->{'Ticket::Frontend::MenuModule'}->{'400-Owner'} =  {
  'Action' => 'AgentTicketOwner',
  'ClusterName' => 'People',
  'ClusterPriority' => '430',
  'Description' => 'Change the owner for this ticket',
  'Link' => 'Action=AgentTicketOwner;TicketID=[% Data.TicketID | html %]',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'Owner',
  'PopupType' => 'TicketAction',
  'Target' => ''
};
```

Ticket::Frontend::MenuModule###410-Responsible

Shows a link in the menu to see the responsible agent of a ticket in the ticket zoom view of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2". To cluster menu items use for Key "ClusterName" and for the Content any name you want to see in the UI. Use "ClusterPriority" to configure the order of a certain cluster within the toolbar.

Default value:

```
$Self->{'Ticket::Frontend::MenuModule'}->{'410-Responsible'} =  {
  'Action' => 'AgentTicketResponsible',
  'ClusterName' => 'People',
  'ClusterPriority' => '430',
```

```
  'Description' => 'Change the responsible for this ticket',
  'Link' => 'Action=AgentTicketResponsible;TicketID=[% Data.TicketID | html %]',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Responsible',
  'Name' => 'Responsible',
  'PopupType' => 'TicketAction',
  'Target' => ''
};
```

Ticket::Frontend::MenuModule###420-Customer

Shows a link in the menu to see the customer who requested the ticket in the ticket zoom view of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2". To cluster menu items use for Key "ClusterName" and for the Content any name you want to see in the UI. Use "ClusterPriority" to configure the order of a certain cluster within the toolbar.

Default value:

```
$Self->{'Ticket::Frontend::MenuModule'}->{'420-Customer'} =  {
  'Action' => 'AgentTicketCustomer',
  'ClusterName' => 'People',
  'ClusterPriority' => '430',
  'Description' => 'Change the customer for this ticket',
  'Link' => 'Action=AgentTicketCustomer;TicketID=[% Data.TicketID | html %]',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'Customer',
  'PopupType' => 'TicketAction',
  'Target' => ''
};
```

Ticket::Frontend::MenuModule###420-Note

Shows a link in the menu to add a note in the ticket zoom view of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2". To cluster menu items use for Key "ClusterName" and for the Content any name you want to see in the UI. Use "ClusterPriority" to configure the order of a certain cluster within the toolbar.

Default value:

```
$Self->{'Ticket::Frontend::MenuModule'}->{'420-Note'} =  {
  'Action' => 'AgentTicketNote',
  'ClusterName' => 'Communication',
  'ClusterPriority' => '435',
  'Description' => 'Add a note to this ticket',
  'Link' => 'Action=AgentTicketNote;TicketID=[% Data.TicketID | html %]',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'Note',
  'PopupType' => 'TicketAction',
  'Target' => ''
};
```

Ticket::Frontend::MenuModule###425-Phone Call Outbound

Shows a link in the menu to add a note in the ticket zoom view of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2". To cluster menu items use for Key "ClusterName" and for the Content any name you want to see in the UI. Use "ClusterPriority" to configure the order of a certain cluster within the toolbar.

Default value:

```
$Self->{'Ticket::Frontend::MenuModule'}->{'425-Phone Call Outbound'} =  {
  'Action' => 'AgentTicketPhoneOutbound',
  'ClusterName' => 'Communication',
  'ClusterPriority' => '435',
  'Description' => 'Add an outbound phone call to this ticket',
  'Link' => 'Action=AgentTicketPhoneOutbound;TicketID=[% Data.TicketID | html %]',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
```

```
  'Name' => 'Phone Call Outbound',
  'PopupType' => 'TicketAction',
  'Target' => ''
};
```

Ticket::Frontend::MenuModule###426-Phone Call Inbound

Shows a link in the menu to add a note in the ticket zoom view of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2". To cluster menu items use for Key "ClusterName" and for the Content any name you want to see in the UI. Use "ClusterPriority" to configure the order of a certain cluster within the toolbar.

Default value:

```
$Self->{'Ticket::Frontend::MenuModule'}->{'426-Phone Call Inbound'} =  {
  'Action' => 'AgentTicketPhoneInbound',
  'ClusterName' => 'Communication',
  'ClusterPriority' => '435',
  'Description' => 'Add an inbound phone call to this ticket',
  'Link' => 'Action=AgentTicketPhoneInbound;TicketID=[% Data.TicketID | html %]',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'Phone Call Inbound',
  'PopupType' => 'TicketAction',
  'Target' => ''
};
```

Ticket::Frontend::MenuModule###427-Email Outbound

Shows a link in the menu to send an outbound email in the ticket zoom view of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2". To cluster menu items use for Key "ClusterName" and for the Content any name you want to see in the UI. Use "ClusterPriority" to configure the order of a certain cluster within the toolbar.

Default value:

```
$Self->{'Ticket::Frontend::MenuModule'}->{'427-Email Outbound'} =  {
  'Action' => 'AgentTicketEmailOutbound',
  'ClusterName' => 'Communication',
  'ClusterPriority' => '435',
  'Description' => 'Send new outgoing mail from this ticket',
  'Link' => 'Action=AgentTicketEmailOutbound;TicketID=[% Data.TicketID | html %]',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'E-Mail Outbound',
  'PopupType' => 'TicketAction',
  'Target' => ''
};
```

Ticket::Frontend::MenuModule###430-Merge

Shows a link in the menu that allows merging tickets in the ticket zoom view of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2". To cluster menu items use for Key "ClusterName" and for the Content any name you want to see in the UI. Use "ClusterPriority" to configure the order of a certain cluster within the toolbar.

Default value:

```
$Self->{'Ticket::Frontend::MenuModule'}->{'430-Merge'} =  {
  'Action' => 'AgentTicketMerge',
  'ClusterName' => 'Miscellaneous',
  'ClusterPriority' => '800',
  'Description' => 'Merge this ticket and all articles into a another ticket',
  'Link' => 'Action=AgentTicketMerge;TicketID=[% Data.TicketID | html %]',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'Merge',
  'PopupType' => 'TicketAction',
  'Target' => ''
};
```

Ticket::Frontend::MenuModule###440-Pending

Shows a link in the menu to set a ticket as pending in the ticket zoom view of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2". To cluster menu items use for Key "ClusterName" and for the Content any name you want to see in the UI. Use "ClusterPriority" to configure the order of a certain cluster within the toolbar.

Default value:

```
$Self->{'Ticket::Frontend::MenuModule'}->{'440-Pending'} =  {
  'Action' => 'AgentTicketPending',
  'ClusterName' => '',
  'ClusterPriority' => '',
  'Description' => 'Set this ticket to pending',
  'Link' => 'Action=AgentTicketPending;TicketID=[% Data.TicketID | html %]',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'Pending',
  'PopupType' => 'TicketAction',
  'Target' => ''
};
```

Ticket::Frontend::MenuModule###448-Watch

Shows a link in the menu for subscribing / unsubscribing from a ticket in the ticket zoom view of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2". To cluster menu items use for Key "ClusterName" and for the Content any name you want to see in the UI. Use "ClusterPriority" to configure the order of a certain cluster within the toolbar.

Default value:

```
$Self->{'Ticket::Frontend::MenuModule'}->{'448-Watch'} =  {
  'Action' => 'AgentTicketWatcher',
  'ClusterName' => '',
  'ClusterPriority' => '',
  'Description' => 'Watch this ticket',
  'Module' => 'Kernel::Output::HTML::TicketMenu::TicketWatcher',
  'Name' => 'Watch',
  'Target' => ''
};
```

Ticket::Frontend::MenuModule###450-Close

Shows a link in the menu to close a ticket in the ticket zoom view of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2". To cluster menu items use for Key "ClusterName" and for the Content any name you want to see in the UI. Use "ClusterPriority" to configure the order of a certain cluster within the toolbar.

Default value:

```
$Self->{'Ticket::Frontend::MenuModule'}->{'450-Close'} =  {
  'Action' => 'AgentTicketClose',
  'ClusterName' => '',
  'ClusterPriority' => '',
  'Description' => 'Close this ticket',
  'Link' => 'Action=AgentTicketClose;TicketID=[% Data.TicketID | html %]',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'Close',
  'PopupType' => 'TicketAction',
  'Target' => ''
};
```

Ticket::Frontend::MenuModule###460-Delete

Shows a link in the menu to delete a ticket in the ticket zoom view of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2". To cluster menu items use

for Key "ClusterName" and for the Content any name you want to see in the UI. Use "ClusterPriority" to configure the order of a certain cluster within the toolbar.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::MenuModule'}->{'460-Delete'} =  {
  'Action' => 'AgentTicketMove',
  'ClusterName' => '',
  'ClusterPriority' => '',
  'Description' => 'Delete this ticket',
  'Link' => 'Action=AgentTicketMove;TicketID=[% Data.TicketID %];DestQueue=Delete',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'Delete',
  'PopupType' => '',
  'Target' => ''
};
```

Ticket::Frontend::MenuModule###470-Junk

Shows a link to set a ticket as junk in the ticket zoom view of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2". To cluster menu items use for Key "ClusterName" and for the Content any name you want to see in the UI. Use "ClusterPriority" to configure the order of a certain cluster within the toolbar.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::MenuModule'}->{'470-Junk'} =  {
  'Action' => 'AgentTicketMove',
  'ClusterName' => '',
  'ClusterPriority' => '',
  'Description' => 'Mark this ticket as junk!',
  'Link' => 'Action=AgentTicketMove;TicketID=[% Data.TicketID %];DestQueue=Junk',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'Spam',
  'PopupType' => '',
  'Target' => ''
};
```

Ticket → Frontend::Agent::Ticket::MenuModulePre

Ticket::Frontend::PreMenuModule###100-Lock

Shows a link in the menu to lock / unlock a ticket in the ticket overviews of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::PreMenuModule'}->{'100-Lock'} =  {
  'Action' => 'AgentTicketLock',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Lock',
  'Name' => 'Lock',
  'PopupType' => '',
  'Target' => ''
};
```

Ticket::Frontend::PreMenuModule###200-Zoom

Shows a link in the menu to zoom a ticket in the ticket overviews of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::PreMenuModule'}->{'200-Zoom'} =  {
  'Action' => 'AgentTicketZoom',
  'Description' => 'Look into a ticket!',
  'Link' => 'Action=AgentTicketZoom;TicketID=[% Data.TicketID | html %]',
```

```
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'Zoom',
  'PopupType' => '',
  'Target' => ''
};
```

Ticket::Frontend::PreMenuModule###210-History

Shows a link in the menu to see the history of a ticket in every ticket overview of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::PreMenuModule'}->{'210-History'} =  {
  'Action' => 'AgentTicketHistory',
  'Description' => 'Show the ticket history',
  'Link' => 'Action=AgentTicketHistory;TicketID=[% Data.TicketID | html %]',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'History',
  'PopupType' => 'TicketHistory',
  'Target' => ''
};
```

Ticket::Frontend::PreMenuModule###300-Priority

Shows a link in the menu to set the priority of a ticket in every ticket overview of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::PreMenuModule'}->{'300-Priority'} =  {
  'Action' => 'AgentTicketPriority',
  'Description' => 'Change the priority for this ticket',
  'Link' => 'Action=AgentTicketPriority;TicketID=[% Data.TicketID | html %]',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'Priority',
  'PopupType' => 'TicketAction',
  'Target' => ''
};
```

Ticket::Frontend::PreMenuModule###420-Note

Shows a link in the menu to add a note to a ticket in every ticket overview of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::PreMenuModule'}->{'420-Note'} =  {
  'Action' => 'AgentTicketNote',
  'Description' => 'Add a note to this ticket',
  'Link' => 'Action=AgentTicketNote;TicketID=[% Data.TicketID | html %]',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'Note',
  'PopupType' => 'TicketAction',
  'Target' => ''
};
```

Ticket::Frontend::PreMenuModule###440-Close

Shows a link in the menu to close a ticket in every ticket overview of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::PreMenuModule'}->{'440-Close'} =  {
  'Action' => 'AgentTicketClose',
  'Description' => 'Close this ticket',
  'Link' => 'Action=AgentTicketClose;TicketID=[% Data.TicketID | html %]',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'Close',
  'PopupType' => 'TicketAction',
  'Target' => ''
};
```

Ticket::Frontend::PreMenuModule###445-Move

Shows a link in the menu to move a ticket in every ticket overview of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::PreMenuModule'}->{'445-Move'} =  {
  'Action' => 'AgentTicketMove',
  'Description' => 'Change queue!',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Move',
  'Name' => 'Move'
};
```

Ticket::Frontend::PreMenuModule###450-Delete

Shows a link in the menu to delete a ticket in every ticket overview of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2".

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::PreMenuModule'}->{'450-Delete'} =  {
  'Action' => 'AgentTicketMove',
  'Description' => 'Delete this ticket',
  'Link' => 'Action=AgentTicketMove;TicketID=[% Data.TicketID %];DestQueue=Delete',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'Delete',
  'PopupType' => '',
  'Target' => ''
};
```

Ticket::Frontend::PreMenuModule###460-Junk

Shows a link in the menu to set a ticket as junk in every ticket overview of the agent interface. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2".

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::PreMenuModule'}->{'460-Junk'} =  {
  'Action' => 'AgentTicketMove',
  'Description' => 'Mark as Spam!',
  'Link' => 'Action=AgentTicketMove;TicketID=[% Data.TicketID %];DestQueue=Junk',
  'Module' => 'Kernel::Output::HTML::TicketMenu::Generic',
  'Name' => 'Spam',
  'PopupType' => '',
  'Target' => ''
};
```

Ticket → Frontend::Agent::Ticket::OverviewMenuModule

Ticket::Frontend::OverviewMenuModule###001-Sort

Shows a select of ticket attributes to order the queue view ticket list. The possible selections can be configured via 'TicketOverviewMenuSort###SortAttributes'.

Default value:

```
$Self->{'Ticket::Frontend::OverviewMenuModule'}->{'001-Sort'} =  {
  'Module' => 'Kernel::Output::HTML::TicketOverviewMenu::Sort'
};
```

TicketOverviewMenuSort###SortAttributes

Defines from which ticket attributes the agent can select the result order.

Default value:

```
$Self->{'TicketOverviewMenuSort'}->{'SortAttributes'} =  {
  'Age' => '1',
  'Title' => '1'
};
```

Ticket → Frontend::Agent::Ticket::ViewBounce

Ticket::Frontend::AgentTicketBounce###Permission
Required permissions to use the ticket bounce screen in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketBounce'}->{'Permission'} =  'bounce';
```

Ticket::Frontend::AgentTicketBounce###RequiredLock
Defines if a ticket lock is required in the ticket bounce screen of the agent interface (if the ticket isn't locked yet, the ticket gets locked and the current agent will be set automatically as its owner).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketBounce'}->{'RequiredLock'} =  '1';
```

Ticket::Frontend::AgentTicketBounce###StateDefault
Defines the default next state of a ticket after being bounced, in the ticket bounce screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketBounce'}->{'StateDefault'} =  'closed
successful';
```

Ticket::Frontend::AgentTicketBounce###StateType
Defines the next state of a ticket after being bounced, in the ticket bounce screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketBounce'}->{'StateType'} =  [
  'open',
  'closed'
];
```

Ticket::Frontend::BounceText
Defines the default ticket bounced notification for customer/sender in the ticket bounce screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::BounceText'} =  'Your email with ticket number
"<OTRS_TICKET>" is bounced to "<OTRS_BOUNCE_TO>". Contact this address for further
information.';
```

Ticket → Frontend::Agent::Ticket::ViewBulk

Ticket::Frontend::AgentTicketBulk###RequiredLock
Automatically lock and set owner to current Agent after selecting for an Bulk Action.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketBulk'}->{'RequiredLock'} =  '1';
```

Ticket::Frontend::AgentTicketBulk###TicketType
Sets the ticket type in the ticket bulk screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketBulk'}->{'TicketType'} =  '1';
```

Ticket::Frontend::AgentTicketBulk###Owner
Sets the ticket owner in the ticket bulk screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketBulk'}->{'Owner'} =  '1';
```

Ticket::Frontend::AgentTicketBulk###Responsible
Sets the responsible agent of the ticket in the ticket bulk screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketBulk'}->{'Responsible'} =  '1';
```

Ticket::Frontend::AgentTicketBulk###State
If a note is added by an agent, sets the state of a ticket in the ticket bulk screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketBulk'}->{'State'} =  '1';
```

Ticket::Frontend::AgentTicketBulk###StateType
Defines the next state of a ticket after adding a note, in the ticket bulk screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketBulk'}->{'StateType'} =  [
  'open',
  'closed',
  'pending reminder',
  'pending auto'
];
```

Ticket::Frontend::AgentTicketBulk###StateDefault
Defines the default next state of a ticket after adding a note, in the ticket bulk screen of the agent interface.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketBulk'}->{'StateDefault'} =  'open';
```

Ticket::Frontend::AgentTicketBulk###Priority
Shows the ticket priority options in the ticket bulk screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketBulk'}->{'Priority'} =  '1';
```

Ticket::Frontend::AgentTicketBulk###PriorityDefault
Defines the default ticket priority in the ticket bulk screen of the agent interface.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketBulk'}->{'PriorityDefault'} =  '3 normal';
```

Ticket::Frontend::AgentTicketBulk###ArticleTypeDefault
Defines the default type of the note in the ticket bulk screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketBulk'}->{'ArticleTypeDefault'} =  'note-
internal';
```

Ticket::Frontend::AgentTicketBulk###ArticleTypes
Specifies the different note types that will be used in the system.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketBulk'}->{'ArticleTypes'} =  {
  'note-external' => '1',
  'note-internal' => '1',
  'note-report' => '0'
};
```

Ticket → Frontend::Agent::Ticket::ViewClose

Ticket::Frontend::AgentTicketClose###Permission
Required permissions to use the close ticket screen in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'Permission'} =  'close';
```

Ticket::Frontend::AgentTicketClose###RequiredLock
Defines if a ticket lock is required in the close ticket screen of the agent interface (if the ticket isn't locked yet, the ticket gets locked and the current agent will be set automatically as its owner).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'RequiredLock'} =  '1';
```

Ticket::Frontend::AgentTicketClose###TicketType
Sets the ticket type in the close ticket screen of the agent interface (Ticket::Type needs to be activated).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'TicketType'} =  '0';
```

Ticket::Frontend::AgentTicketClose###Service
Sets the service in the close ticket screen of the agent interface (Ticket::Service needs to be activated).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'Service'} =  '0';
```

Ticket::Frontend::AgentTicketClose###ServiceMandatory
Sets if service must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'ServiceMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketClose###SLAMandatory
Sets if SLA must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'SLAMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketClose###Queue
Sets the queue in the ticket close screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'Queue'} =  '0';
```

Ticket::Frontend::AgentTicketClose###Owner
Sets the ticket owner in the close ticket screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'Owner'} =  '0';
```

Ticket::Frontend::AgentTicketClose###OwnerMandatory
Sets if ticket owner must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'OwnerMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketClose###Responsible
Sets the responsible agent of the ticket in the close ticket screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'Responsible'} =  '0';
```

Ticket::Frontend::AgentTicketClose###State
If a note is added by an agent, sets the state of a ticket in the close ticket screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'State'} =  '1';
```

Ticket::Frontend::AgentTicketClose###StateType
Defines the next state of a ticket after adding a note, in the close ticket screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'StateType'} =  [
  'closed'
];
```

Ticket::Frontend::AgentTicketClose###StateDefault
Defines the default next state of a ticket after adding a note, in the close ticket screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'StateDefault'} =  'closed successful';
```

Ticket::Frontend::AgentTicketClose###Note
Allows adding notes in the close ticket screen of the agent interface. Can be overwritten by Ticket::Frontend::NeedAccountedTime.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'Note'} =  '1';
```

Ticket::Frontend::AgentTicketClose###NoteMandatory
Sets if note must be filled in by the agent. Can be overwritten by Ticket::Frontend::NeedAccountedTime.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'NoteMandatory'} =  '1';
```

Ticket::Frontend::AgentTicketClose###Subject
Sets the default subject for notes added in the close ticket screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'Subject'} =  '';
```

Ticket::Frontend::AgentTicketClose###Body
Sets the default body text for notes added in the close ticket screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'Body'} =  '';
```

Ticket::Frontend::AgentTicketClose###InvolvedAgent
Shows a list of all the involved agents on this ticket, in the close ticket screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'InvolvedAgent'} =  '0';
```

Ticket::Frontend::AgentTicketClose###InformAgent
Shows a list of all the possible agents (all agents with note permissions on the queue/ticket) to determine who should be informed about this note, in the close ticket screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'InformAgent'} =  '0';
```

Ticket::Frontend::AgentTicketClose###ArticleTypeDefault
Defines the default type of the note in the close ticket screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'ArticleTypeDefault'} =  'note-
internal';
```

Ticket::Frontend::AgentTicketClose###ArticleTypes
Specifies the available note types for this ticket mask. If the option is deselected, ArticleTypeDefault is used and the option is removed from the mask.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'ArticleTypes'} =  {
 'note-external' => '0',
 'note-internal' => '1',
```

```
  'note-report' => '0'
};
```

Ticket::Frontend::AgentTicketClose###Priority
Shows the ticket priority options in the close ticket screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'Priority'} =  '0';
```

Ticket::Frontend::AgentTicketClose###PriorityDefault
Defines the default ticket priority in the close ticket screen of the agent interface.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'PriorityDefault'} =  '3 normal';
```

Ticket::Frontend::AgentTicketClose###Title
Shows the title fields in the close ticket screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'Title'} =  '0';
```

Ticket::Frontend::AgentTicketClose###HistoryType
Defines the history type for the close ticket screen action, which gets used for ticket history in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'HistoryType'} =  'AddNote';
```

Ticket::Frontend::AgentTicketClose###HistoryComment
Defines the history comment for the close ticket screen action, which gets used for ticket history in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'HistoryComment'} =  '%%Close';
```

Ticket::Frontend::AgentTicketClose###DynamicField
Dynamic fields shown in the ticket close screen of the agent interface. Possible settings: 0 = Disabled, 1 = Enabled, 2 = Enabled and required.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'DynamicField'} =  {};
```

Ticket::Frontend::AgentTicketClose###RichTextWidth
Defines the width for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'RichTextWidth'} =  '620';
```

Ticket::Frontend::AgentTicketClose###RichTextHeight
Defines the height for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketClose'}->{'RichTextHeight'} =  '100';
```

Ticket → Frontend::Agent::Ticket::ViewCompose

Ticket::Frontend::AgentTicketCompose###Permission
Required permissions to use the ticket compose screen in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketCompose'}->{'Permission'} =  'compose';
```

Ticket::Frontend::AgentTicketCompose###RequiredLock
Defines if a ticket lock is required in the ticket compose screen of the agent interface (if the ticket isn't locked yet, the ticket gets locked and the current agent will be set automatically as its owner).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketCompose'}->{'RequiredLock'} =  '1';
```

Ticket::Frontend::AgentTicketCompose###StateDefault
Defines the default next state of a ticket if it is composed / answered in the ticket compose screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketCompose'}->{'StateDefault'} =  'open';
```

Ticket::Frontend::AgentTicketCompose###StateType
Defines the next possible states after composing / answering a ticket in the ticket compose screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketCompose'}->{'StateType'} =  [
  'open',
  'closed',
  'pending auto',
  'pending reminder'
];
```

Ticket::Frontend::AgentTicketCompose###ArticleTypes
Specifies the different article types that will be used in the system.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketCompose'}->{'ArticleTypes'} =  [
  'email-external',
  'email-internal'
];
```

Ticket::Frontend::AgentTicketCompose###DefaultArticleType
Specifies the default article type for the ticket compose screen in the agent interface if the article type cannot be automatically detected.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketCompose'}->{'DefaultArticleType'} =  'email-
external';
```

Ticket::Frontend::ResponseFormat

Defines the format of responses in the ticket compose screen of the agent interface ([% Data.OrigFrom | html %] is From 1:1, [% Data.OrigFromName | html %] is only realname of From).

This setting can not be deactivated.

Default value:

```
 $Self->{'Ticket::Frontend::ResponseFormat'} =  '[% Data.Salutation | html %]
[% Data.StdResponse | html %]
[% Data.Signature | html %]

[% Data.Created | Localize("TimeShort") %] - [% Data.OrigFromName | html %] [%
 Translate("wrote") | html %]:
[% Data.Body | html %]
';
```

Ticket::Frontend::Quote

Defines the used character for plaintext email quotes in the ticket compose screen of the agent interface. If this is empty or inactive, original emails will not be quoted but appended to the response.

Default value:

```
 $Self->{'Ticket::Frontend::Quote'} =  '>';
```

Ticket::Frontend::ResponseQuoteMaxLines

Defines the maximum number of quoted lines to be added to responses.

This setting is not active by default.

Default value:

```
 $Self->{'Ticket::Frontend::ResponseQuoteMaxLines'} =  '99';
```

Ticket::Frontend::ComposeAddCustomerAddress

Adds customers email addresses to recipients in the ticket compose screen of the agent interface. The customers email address won't be added if the article type is email-internal.

This setting can not be deactivated.

Default value:

```
 $Self->{'Ticket::Frontend::ComposeAddCustomerAddress'} =  '1';
```

Ticket::Frontend::ComposeReplaceSenderAddress

Replaces the original sender with current customer's email address on compose answer in the ticket compose screen of the agent interface.

This setting can not be deactivated.

Default value:

```
 $Self->{'Ticket::Frontend::ComposeReplaceSenderAddress'} =  '0';
```

Ticket::Frontend::ComposeExcludeCcRecipients

Uses Cc recipients in reply Cc list on compose an email answer in the ticket compose screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::ComposeExcludeCcRecipients'} =  '0';
```

Ticket::Frontend::AgentTicketCompose###DynamicField
Dynamic fields shown in the ticket compose screen of the agent interface. Possible settings: 0 = Disabled, 1 = Enabled, 2 = Enabled and required.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketCompose'}->{'DynamicField'} =  {};
```

Ticket → Frontend::Agent::Ticket::ViewCustomer

Ticket::Frontend::AgentTicketCustomer###Permission
Required permissions to change the customer of a ticket in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketCustomer'}->{'Permission'} =  'customer';
```

Ticket::Frontend::AgentTicketCustomer###RequiredLock
Defines if a ticket lock is required to change the customer of a ticket in the agent interface (if the ticket isn't locked yet, the ticket gets locked and the current agent will be set automatically as its owner).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketCustomer'}->{'RequiredLock'} =  '0';
```

Ticket::Frontend::AgentTicketCustomer::CustomerIDReadOnly
Controls if CutomerID is editable in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketCustomer::CustomerIDReadOnly'} =  '1';
```

Ticket → Frontend::Agent::Ticket::ViewEmailNew

Ticket::Frontend::AgentTicketEmail###Priority
Sets the default priority for new email tickets in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmail'}->{'Priority'} =  '3 normal';
```

Ticket::Frontend::AgentTicketEmail###ArticleType
Sets the default article type for new email tickets in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmail'}->{'ArticleType'} =  'email-external';
```

Ticket::Frontend::AgentTicketEmail###SenderType
Sets the default sender type for new email tickets in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmail'}->{'SenderType'} =  'agent';
```

Ticket::Frontend::AgentTicketEmail::CustomerIDReadOnly
Controls if CutomerID is editable in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmail::CustomerIDReadOnly'} =  '1';
```

Ticket::Frontend::AgentTicketEmail###Subject
Sets the default subject for new email tickets (e.g. 'email Outbound') in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmail'}->{'Subject'} =  '';
```

Ticket::Frontend::AgentTicketEmail###Body
Sets the default text for new email tickets in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmail'}->{'Body'} =  '';
```

Ticket::Frontend::AgentTicketEmail###StateDefault
Sets the default next ticket state, after the creation of an email ticket in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmail'}->{'StateDefault'} =  'open';
```

Ticket::Frontend::AgentTicketEmail###StateType
Determines the next possible ticket states, after the creation of a new email ticket in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmail'}->{'StateType'} =  [
  'open',
  'pending auto',
  'pending reminder',
  'closed'
];
```

Ticket::Frontend::AgentTicketEmail###HistoryType
Defines the history type for the email ticket screen action, which gets used for ticket history in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmail'}->{'HistoryType'} =  'EmailAgent';
```

Ticket::Frontend::AgentTicketEmail###HistoryComment
Defines the history comment for the email ticket screen action, which gets used for ticket history in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmail'}->{'HistoryComment'} =  '';
```

Ticket::Frontend::AgentTicketEmail###ServiceMandatory
Sets if service must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmail'}->{'ServiceMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketEmail###SLAMandatory
Sets if SLA must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmail'}->{'SLAMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketEmail###DynamicField
Dynamic fields shown in the ticket email screen of the agent interface. Possible settings: 0 = Disabled, 1 = Enabled, 2 = Enabled and required.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmail'}->{'DynamicField'} =  {};
```

Ticket::Frontend::AgentTicketEmail###RichTextWidth
Defines the width for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmail'}->{'RichTextWidth'} =  '620';
```

Ticket::Frontend::AgentTicketEmail###RichTextHeight
Defines the height for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmail'}->{'RichTextHeight'} =  '320';
```

Ticket → Frontend::Agent::Ticket::ViewEmailOutbound

Ticket::Frontend::AgentTicketEmailOutbound###Permission
Required permissions to use the email outbound screen in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmailOutbound'}->{'Permission'} =  'compose';
```

Ticket::Frontend::AgentTicketEmailOutbound###RequiredLock
Defines if a ticket lock is required in the email outbound screen of the agent interface (if the ticket isn't locked yet, the ticket gets locked and the current agent will be set automatically as its owner).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmailOutbound'}->{'RequiredLock'} =  '1';
```

Ticket::Frontend::AgentTicketEmailOutbound###StateDefault
Defines the default next state of a ticket after the message has been sent, in the email outbound screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmailOutbound'}->{'StateDefault'} =  'open';
```

Ticket::Frontend::AgentTicketEmailOutbound###StateType
Defines the next possible states after sending a message in the email outbound screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmailOutbound'}->{'StateType'} =  [
  'open',
  'closed',
  'pending reminder',
  'pending auto'
];
```

Ticket::Frontend::AgentTicketEmailOutbound###ArticleTypeDefault
Defines the default type of the message in the email outbound screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmailOutbound'}->{'ArticleTypeDefault'} =
'email-internal';
```

Ticket::Frontend::AgentTicketEmailOutbound###ArticleTypes
Specifies the different article types that will be used in the system.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmailOutbound'}->{'ArticleTypes'} =  [
  'email-external',
  'email-internal'
];
```

Ticket::Frontend::AgentTicketEmailOutbound###DynamicField
Dynamic fields shown in the email outbound screen of the agent interface. Possible settings: 0 = Disabled, 1 = Enabled, 2 = Enabled and required.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmailOutbound'}->{'DynamicField'} =  {};
```

Ticket::Frontend::AgentTicketEmailOutbound###RichTextWidth
Defines the width for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmailOutbound'}->{'RichTextWidth'} =  '620';
```

Ticket::Frontend::AgentTicketEmailOutbound###RichTextHeight
Defines the height for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEmailOutbound'}->{'RichTextHeight'} =  '300';
```

Ticket → Frontend::Agent::Ticket::ViewEscalation

Ticket::Frontend::AgentTicketEscalationView###TicketPermission
Defines the required permission to show a ticket in the escalation view of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEscalationView'}->{'TicketPermission'} =  'rw';
```

Ticket::Frontend::AgentTicketEscalationView###ViewableTicketsPage
Shows all open tickets (even if they are locked) in the escalation view of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEscalationView'}->{'ViewableTicketsPage'} =
'50';
```

Ticket::Frontend::AgentTicketEscalationView###SortBy::Default
Defines the default ticket attribute for ticket sorting in the escalation view of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEscalationView'}->{'SortBy::Default'} =
'EscalationTime';
```

Ticket::Frontend::AgentTicketEscalationView###Order::Default
Defines the default ticket order (after priority sort) in the escalation view of the agent interface. Up: oldest on top. Down: latest on top.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEscalationView'}->{'Order::Default'} =  'Up';
```

Ticket::Frontend::AgentTicketEscalationView###DefaultColumns
Columns that can be filtered in the escalation view of the agent interface. Possible settings: 0 = Disabled, 1 = Available, 2 = Enabled by default. Note: Only Ticket attributes, Dynamic Fields (DynamicField_NameX) and Customer attributes (e.g. CustomerUserPhone, CustomerCompanyName, ...) are allowed.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketEscalationView'}->{'DefaultColumns'} =  {
 'Age' => '2',
 'Changed' => '1',
 'Created' => '1',
 'CustomerCompanyName' => '1',
 'CustomerID' => '2',
 'CustomerName' => '1',
 'CustomerUserID' => '1',
 'EscalationResponseTime' => '1',
 'EscalationSolutionTime' => '1',
 'EscalationTime' => '2',
 'EscalationUpdateTime' => '1',
 'Lock' => '2',
```

```
  'Owner' => '2',
  'PendingTime' => '1',
  'Priority' => '1',
  'Queue' => '2',
  'Responsible' => '1',
  'SLA' => '1',
  'Service' => '1',
  'State' => '2',
  'TicketNumber' => '2',
  'Title' => '2',
  'Type' => '1'
};
```

Ticket → Frontend::Agent::Ticket::ViewForward

Ticket::Frontend::AgentTicketForward###Permission
Required permissions to use the ticket forward screen in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketForward'}->{'Permission'} =  'forward';
```

Ticket::Frontend::AgentTicketForward###RequiredLock
Defines if a ticket lock is required in the ticket forward screen of the agent interface (if the ticket isn't locked yet, the ticket gets locked and the current agent will be set automatically as its owner).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketForward'}->{'RequiredLock'} =  '1';
```

Ticket::Frontend::AgentTicketForward###StateDefault
Defines the default next state of a ticket after being forwarded, in the ticket forward screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketForward'}->{'StateDefault'} =  'closed
successful';
```

Ticket::Frontend::AgentTicketForward###StateType
Defines the next possible states after forwarding a ticket in the ticket forward screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketForward'}->{'StateType'} =  [
  'open',
  'closed',
  'pending reminder',
  'pending auto'
];
```

Ticket::Frontend::AgentTicketForward###ArticleTypeDefault
Defines the default type of forwarded message in the ticket forward screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketForward'}->{'ArticleTypeDefault'} =  'email-
external';
```

Ticket::Frontend::AgentTicketForward###ArticleTypes
Specifies the different article types that will be used in the system.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketForward'}->{'ArticleTypes'} =  [
  'email-external',
  'email-internal'
];
```

Ticket::Frontend::AgentTicketForward###DynamicField
Dynamic fields shown in the ticket forward screen of the agent interface. Possible settings: 0 = Disabled, 1 = Enabled, 2 = Enabled and required.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketForward'}->{'DynamicField'} =  {};
```

Ticket::Frontend::AgentTicketForward###RichTextWidth
Defines the width for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketForward'}->{'RichTextWidth'} =  '620';
```

Ticket::Frontend::AgentTicketForward###RichTextHeight
Defines the height for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketForward'}->{'RichTextHeight'} =  '100';
```

Ticket → Frontend::Agent::Ticket::ViewFreeText

Ticket::Frontend::AgentTicketFreeText###Permission
Required permissions to use the ticket free text screen in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'Permission'} =  'rw';
```

Ticket::Frontend::AgentTicketFreeText###RequiredLock
Defines if a ticket lock is required in the ticket free text screen of the agent interface (if the ticket isn't locked yet, the ticket gets locked and the current agent will be set automatically as its owner).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'RequiredLock'} =  '0';
```

Ticket::Frontend::AgentTicketFreeText###TicketType
Sets the ticket type in the ticket free text screen of the agent interface (Ticket::Type needs to be activated).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'TicketType'} =  '1';
```

Ticket::Frontend::AgentTicketFreeText###Service
Sets the service in the ticket free text screen of the agent interface (Ticket::Service needs to be activated).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'Service'} =  '1';
```

Ticket::Frontend::AgentTicketFreeText###ServiceMandatory
Sets if service must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'ServiceMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketFreeText###SLAMandatory
Sets if SLA must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'SLAMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketFreeText###Queue
Sets the queue in the ticket free text screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'Queue'} =  '0';
```

Ticket::Frontend::AgentTicketFreeText###Owner
Sets the ticket owner in the ticket free text screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'Owner'} =  '0';
```

Ticket::Frontend::AgentTicketFreeText###OwnerMandatory
Sets if ticket owner must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'OwnerMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketFreeText###Responsible
Sets the responsible agent of the ticket in the ticket free text screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'Responsible'} =  '0';
```

Ticket::Frontend::AgentTicketFreeText###State
If a note is added by an agent, sets the state of a ticket in the ticket free text screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'State'} =  '0';
```

Ticket::Frontend::AgentTicketFreeText###StateType
Defines the next state of a ticket after adding a note, in the ticket free text screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'StateType'} =  [
 'open',
 'closed',
 'pending reminder',
```

```
  'pending auto'
];
```

Ticket::Frontend::AgentTicketFreeText###StateDefault
Defines the default next state of a ticket after adding a note, in the ticket free text screen of the agent interface.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'StateDefault'} =  'open';
```

Ticket::Frontend::AgentTicketFreeText###Note
Allows adding notes in the ticket free text screen of the agent interface. Can be overwritten by Ticket::Frontend::NeedAccountedTime.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'Note'} =  '0';
```

Ticket::Frontend::AgentTicketFreeText###NoteMandatory
Sets if note must be filled in by the agent. Can be overwritten by Ticket::Frontend::NeedAccountedTime.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'NoteMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketFreeText###Subject
Defines the default subject of a note in the ticket free text screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'Subject'} =  '';
```

Ticket::Frontend::AgentTicketFreeText###Body
Defines the default body of a note in the ticket free text screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'Body'} =  '';
```

Ticket::Frontend::AgentTicketFreeText###InvolvedAgent
Shows a list of all the involved agents on this ticket, in the ticket free text screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'InvolvedAgent'} =  '0';
```

Ticket::Frontend::AgentTicketFreeText###InformAgent
Shows a list of all the possible agents (all agents with note permissions on the queue/ticket) to determine who should be informed about this note, in the ticket free text screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'InformAgent'} =  '0';
```

Ticket::Frontend::AgentTicketFreeText###ArticleTypeDefault
Defines the default type of the note in the ticket free text screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'ArticleTypeDefault'} =  'note-internal';
```

Ticket::Frontend::AgentTicketFreeText###ArticleTypes
Specifies the available note types for this ticket mask. If the option is deselected, ArticleTypeDefault is used and the option is removed from the mask.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'ArticleTypes'} =  {
  'note-external' => '1',
  'note-internal' => '1',
  'note-report' => '0'
};
```

Ticket::Frontend::AgentTicketFreeText###Priority
Shows the ticket priority options in the ticket free text screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'Priority'} =  '0';
```

Ticket::Frontend::AgentTicketFreeText###PriorityDefault
Defines the default ticket priority in the ticket free text screen of the agent interface.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'PriorityDefault'} =  '3 normal';
```

Ticket::Frontend::AgentTicketFreeText###Title
Shows the title field in the ticket free text screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'Title'} =  '1';
```

Ticket::Frontend::AgentTicketFreeText###HistoryType
Defines the history type for the ticket free text screen action, which gets used for ticket history.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'HistoryType'} =  'AddNote';
```

Ticket::Frontend::AgentTicketFreeText###HistoryComment
Defines the history comment for the ticket free text screen action, which gets used for ticket history.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'HistoryComment'} =  '%%FreeText';
```

Ticket::Frontend::AgentTicketFreeText###DynamicField
Dynamic fields shown in the ticket free text screen of the agent interface. Possible settings: 0 = Disabled, 1 = Enabled, 2 = Enabled and required.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'DynamicField'} =  {};
```

Ticket::Frontend::AgentTicketFreeText###RichTextWidth
Defines the width for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'RichTextWidth'} =  '620';
```

Ticket::Frontend::AgentTicketFreeText###RichTextHeight
Defines the height for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketFreeText'}->{'RichTextHeight'} =  '100';
```

Ticket → Frontend::Agent::Ticket::ViewHistory

Ticket::Frontend::HistoryOrder
Shows the ticket history (reverse ordered) in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::HistoryOrder'} =  'normal';
```

Ticket::Frontend::HistoryTypes###000-Framework
Controls how to display the ticket history entries as readable values.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::HistoryTypes'}->{'000-Framework'} =  {
 'AddNote' => 'Added note (%s)',
 'ArchiveFlagUpdate' => 'Archive state changed: "%s"',
 'Bounce' => 'Bounced to "%s".',
 'CustomerUpdate' => 'Updated: %s',
 'EmailAgent' => 'Email sent to customer.',
 'EmailCustomer' => 'Added email. %s',
 'EscalationResponseTimeNotifyBefore' => 'Escalation response time forewarned',
 'EscalationResponseTimeStart' => 'Escalation response time in effect',
 'EscalationResponseTimeStop' => 'Escalation response time finished',
 'EscalationSolutionTimeNotifyBefore' => 'Escalation solution time forewarned',
 'EscalationSolutionTimeStart' => 'Escalation solution time in effect',
 'EscalationSolutionTimeStop' => 'Escalation solution time finished',
 'EscalationUpdateTimeNotifyBefore' => 'Escalation update time forewarned',
 'EscalationUpdateTimeStart' => 'Escalation update time in effect',
 'EscalationUpdateTimeStop' => 'Escalation update time finished',
 'FollowUp' => 'FollowUp for [%s]. %s',
 'Forward' => 'Forwarded to "%s".',
 'Lock' => 'Locked ticket.',
 'LoopProtection' => 'Loop-Protection! No auto-response sent to "%s".',
 'Misc' => '%s',
 'Move' => 'Ticket moved into Queue "%s" (%s) from Queue "%s" (%s).',
 'NewTicket' => 'New Ticket [%s] created (Q=%s;P=%s;S=%s).',
 'OwnerUpdate' => 'New owner is "%s" (ID=%s).',
 'PhoneCallAgent' => 'Agent called customer.',
 'PhoneCallCustomer' => 'Customer called us.',
 'PriorityUpdate' => 'Changed priority from "%s" (%s) to "%s" (%s).',
 'Remove' => '%s',
 'ResponsibleUpdate' => 'New responsible is "%s" (ID=%s).',
 'SLAUpdate' => 'Updated SLA to %s (ID=%s).',
```

```
  'SendAgentNotification' => '"%s" notification was sent to "%s" by "%s".',
  'SendAnswer' => 'Email sent to "%s".',
  'SendAutoFollowUp' => 'AutoFollowUp sent to "%s".',
  'SendAutoReject' => 'AutoReject sent to "%s".',
  'SendAutoReply' => 'AutoReply sent to "%s".',
  'SendCustomerNotification' => 'Notification sent to "%s".',
  'ServiceUpdate' => 'Updated Service to %s (ID=%s).',
  'SetPendingTime' => 'Updated: %s',
  'StateUpdate' => 'Old: "%s" New: "%s"',
  'Subscribe' => 'Added subscription for user "%s".',
  'SystemRequest' => 'System Request (%s).',
  'TicketDynamicFieldUpdate' => 'Updated: %s=%s;%s=%s;%s=%s;',
  'TicketLinkAdd' => 'Added link to ticket "%s".',
  'TicketLinkDelete' => 'Deleted link to ticket "%s".',
  'TimeAccounting' => '%s time unit(s) accounted. Now total %s time unit(s).',
  'TitleUpdate' => 'Title updated: Old: "%s", New: "%s"',
  'TypeUpdate' => 'Updated Type to %s (ID=%s).',
  'Unlock' => 'Unlocked ticket.',
  'Unsubscribe' => 'Removed subscription for user "%s".',
  'WebRequestCustomer' => 'Customer request via web.'
};
```

Ticket → Frontend::Agent::Ticket::ViewLocked

Ticket::Frontend::AgentTicketLockedView###SortBy::Default
Defines the default ticket attribute for ticket sorting in the locked ticket view of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketLockedView'}->{'SortBy::Default'} =  'Age';
```

Ticket::Frontend::AgentTicketLockedView###Order::Default
Defines the default ticket order in the ticket locked view of the agent interface. Up: oldest on top. Down: latest on top.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketLockedView'}->{'Order::Default'} =  'Up';
```

Ticket::Frontend::AgentTicketLockedView###DefaultColumns
Columns that can be filtered in the locked view of the agent interface. Possible settings: 0 = Disabled, 1 = Available, 2 = Enabled by default. Note: Only Ticket attributes, Dynamic Fields (DynamicField_NameX) and Customer attributes (e.g. CustomerUserPhone, CustomerCompanyName, ...) are allowed.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketLockedView'}->{'DefaultColumns'} =  {
 'Age' => '2',
 'Changed' => '1',
 'Created' => '1',
 'CustomerCompanyName' => '1',
 'CustomerID' => '2',
 'CustomerName' => '1',
 'CustomerUserID' => '1',
 'EscalationResponseTime' => '1',
 'EscalationSolutionTime' => '1',
 'EscalationTime' => '1',
 'EscalationUpdateTime' => '1',
 'Lock' => '2',
 'Owner' => '2',
 'PendingTime' => '1',
 'Priority' => '1',
```

```
  'Queue' => '2',
  'Responsible' => '1',
  'SLA' => '1',
  'Service' => '1',
  'State' => '2',
  'TicketNumber' => '2',
  'Title' => '2',
  'Type' => '1'
};
```

Ticket → Frontend::Agent::Ticket::ViewMerge

Ticket::Frontend::AgentTicketMerge###Permission
Required permissions to use the ticket merge screen of a zoomed ticket in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketMerge'}->{'Permission'} =  'rw';
```

Ticket::Frontend::AgentTicketMerge###RequiredLock
Defines if a ticket lock is required in the ticket merge screen of a zoomed ticket in the agent interface (if the ticket isn't locked yet, the ticket gets locked and the current agent will be set automatically as its owner).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketMerge'}->{'RequiredLock'} =  '1';
```

Ticket::Frontend::MergeText
When tickets are merged, the customer can be informed per email by setting the check box "Inform Sender". In this text area, you can define a pre-formatted text which can later be modified by the agents.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::MergeText'} =  'Your email with ticket number
"<OTRS_TICKET>" is merged to "<OTRS_MERGE_TO_TICKET>".';
```

Ticket::Frontend::AutomaticMergeSubject
When tickets are merged, a note will be added automatically to the ticket which is no longer active. Here you can define the subject of this note (this subject cannot be changed by the agent).

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AutomaticMergeSubject'} =  'Ticket Merged';
```

Ticket::Frontend::AutomaticMergeText
When tickets are merged, a note will be added automatically to the ticket which is no longer active. Here you can define the body of this note (this text cannot be changed by the agent).

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AutomaticMergeText'} =  'Merged Ticket <OTRS_TICKET> to
<OTRS_MERGE_TO_TICKET>.';
```

Ticket::Frontend::AgentTicketMerge###RichTextWidth
Defines the width for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketMerge'}->{'RichTextWidth'} =  '620';
```

Ticket::Frontend::AgentTicketMerge###RichTextHeight
Defines the height for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketMerge'}->{'RichTextHeight'} =  '100';
```

Ticket → Frontend::Agent::Ticket::ViewMove

Ticket::Frontend::MoveType
Determines if the list of possible queues to move to ticket into should be displayed in a dropdown list or in a new window in the agent interface. If "New Window" is set you can add a move note to the ticket.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::MoveType'} =  'form';
```

Ticket::Frontend::AgentTicketMove###RequiredLock
Automatically lock and set owner to current Agent after opening the move ticket screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketMove'}->{'RequiredLock'} =  '1';
```

Ticket::Frontend::AgentTicketMove###State
Allows to set a new ticket state in the move ticket screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketMove'}->{'State'} =  '1';
```

Ticket::Frontend::AgentTicketMove###StateType
Defines the next state of a ticket after being moved to another queue, in the move ticket screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketMove'}->{'StateType'} =  [
  'open',
  'closed'
];
```

Ticket::Frontend::AgentTicketMove###Priority
Shows the ticket priority options in the move ticket screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketMove'}->{'Priority'} =  '0';
```

Ticket::Frontend::AgentTicketMove###Note

Allows adding notes in the ticket free text screen of the agent interface. Can be overwritten by Ticket::Frontend::NeedAccountedTime.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketMove'}->{'Note'} =  '0';
```

Ticket::Frontend::AgentTicketMove###NoteMandatory

Sets if note must be filled in by the agent. Can be overwritten by Ticket::Frontend::NeedAccountedTime.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketMove'}->{'NoteMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketMove###NextScreen

Determines the next screen after the ticket is moved. LastScreenOverview will return the last overview screen (e.g. search results, queueview, dashboard). TicketZoom will return to the TicketZoom.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketMove'}->{'NextScreen'} =  'TicketZoom';
```

Ticket::Frontend::AgentTicketMove###Subject

Sets the default subject for notes added in the ticket move screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketMove'}->{'Subject'} =  '';
```

Ticket::Frontend::AgentTicketMove###Body

Sets the default body text for notes added in the ticket move screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketMove'}->{'Body'} =  '';
```

Ticket::Frontend::AgentTicketMove###RichTextWidth

Defines the width for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketMove'}->{'RichTextWidth'} =  '620';
```

Ticket::Frontend::AgentTicketMove###RichTextHeight

Defines the height for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketMove'}->{'RichTextHeight'} =  '100';
```

Ticket::Frontend::AgentTicketMove###DynamicField

Dynamic fields shown in the ticket move screen of the agent interface. Possible settings: 0 = Disabled, 1 = Enabled, 2 = Enabled and required.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketMove'}->{'DynamicField'} =  {};
```

Ticket → Frontend::Agent::Ticket::ViewNote

Ticket::Frontend::AgentTicketNote###Permission
Required permissions to use the ticket note screen in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'Permission'} =  'note';
```

Ticket::Frontend::AgentTicketNote###RequiredLock
Defines if a ticket lock is required in the ticket note screen of the agent interface (if the ticket isn't locked yet, the ticket gets locked and the current agent will be set automatically as its owner).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'RequiredLock'} =  '0';
```

Ticket::Frontend::AgentTicketNote###TicketType
Sets the ticket type in the ticket note screen of the agent interface (Ticket::Type needs to be activated).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'TicketType'} =  '0';
```

Ticket::Frontend::AgentTicketNote###Service
Sets the service in the ticket note screen of the agent interface (Ticket::Service needs to be activated).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'Service'} =  '0';
```

Ticket::Frontend::AgentTicketNote###ServiceMandatory
Sets if service must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'ServiceMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketNote###SLAMandatory
Sets if SLA must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'SLAMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketNote###Queue
Sets the queue in the ticket note screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'Queue'} =  '0';
```

Ticket::Frontend::AgentTicketNote###Owner
Sets the ticket owner in the ticket note screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'Owner'} =  '0';
```

Ticket::Frontend::AgentTicketNote###OwnerMandatory
Sets if ticket owner must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'OwnerMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketNote###Responsible
Sets the responsible agent of the ticket in the ticket note screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'Responsible'} =  '0';
```

Ticket::Frontend::AgentTicketNote###State
If a note is added by an agent, sets the state of a ticket in the ticket note screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'State'} =  '0';
```

Ticket::Frontend::AgentTicketNote###StateType
Defines the next state of a ticket after adding a note, in the ticket note screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'StateType'} =  [
  'open',
  'closed',
  'pending reminder',
  'pending auto'
];
```

Ticket::Frontend::AgentTicketNote###StateDefault
Defines the default next state of a ticket after adding a note, in the ticket note screen of the agent interface.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'StateDefault'} =  'open';
```

Ticket::Frontend::AgentTicketNote###Note
Allows adding notes in the ticket note screen of the agent interface. Can be overwritten by Ticket::Frontend::NeedAccountedTime.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'Note'} =  '1';
```

Ticket::Frontend::AgentTicketNote###NoteMandatory
Sets if note must be filled in by the agent. Can be overwritten by Ticket::Frontend::NeedAccountedTime.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'NoteMandatory'} =  '1';
```

Ticket::Frontend::AgentTicketNote###Subject
Sets the default subject for notes added in the ticket note screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'Subject'} =  '';
```

Ticket::Frontend::AgentTicketNote###Body
Sets the default body text for notes added in the ticket note screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'Body'} =  '';
```

Ticket::Frontend::AgentTicketNote###InvolvedAgent
Shows a list of all the involved agents on this ticket, in the ticket note screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'InvolvedAgent'} =  '0';
```

Ticket::Frontend::AgentTicketNote###InformAgent
Shows a list of all the possible agents (all agents with note permissions on the queue/ticket) to determine who should be informed about this note, in the ticket note screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'InformAgent'} =  '0';
```

Ticket::Frontend::AgentTicketNote###ArticleTypeDefault
Defines the default type of the note in the ticket note screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'ArticleTypeDefault'} =  'note-internal';
```

Ticket::Frontend::AgentTicketNote###ArticleTypes
Specifies the available note types for this ticket mask. If the option is deselected, ArticleTypeDefault is used and the option is removed from the mask.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'ArticleTypes'} =  {
  'note-external' => '1',
  'note-internal' => '1',
  'note-report' => '0'
};
```

Ticket::Frontend::AgentTicketNote###Priority
Shows the ticket priority options in the ticket note screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'Priority'} =  '0';
```

Ticket::Frontend::AgentTicketNote###PriorityDefault
Defines the default ticket priority in the ticket note screen of the agent interface.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'PriorityDefault'} =  '3 normal';
```

Ticket::Frontend::AgentTicketNote###Title
Shows the title fields in the ticket note screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'Title'} =  '0';
```

Ticket::Frontend::AgentTicketNote###HistoryType
Defines the history type for the ticket note screen action, which gets used for ticket history in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'HistoryType'} =  'AddNote';
```

Ticket::Frontend::AgentTicketNote###HistoryComment
Defines the history comment for the ticket note screen action, which gets used for ticket history in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'HistoryComment'} =  '%%Note';
```

Ticket::Frontend::AgentTicketNote###DynamicField
Dynamic fields shown in the ticket note screen of the agent interface. Possible settings: 0 = Disabled, 1 = Enabled, 2 = Enabled and required.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'DynamicField'} =  {};
```

Ticket::Frontend::AgentTicketNote###RichTextWidth
Defines the width for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'RichTextWidth'} =  '620';
```

Ticket::Frontend::AgentTicketNote###RichTextHeight
Defines the height for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketNote'}->{'RichTextHeight'} =  '100';
```

Ticket → Frontend::Agent::Ticket::ViewOwner

Ticket::Frontend::AgentTicketOwner###Permission
Required permissions to use the ticket owner screen of a zoomed ticket in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'Permission'} =  'owner';
```

Ticket::Frontend::AgentTicketOwner###RequiredLock
Defines if a ticket lock is required in the ticket owner screen of a zoomed ticket in the agent interface (if the ticket isn't locked yet, the ticket gets locked and the current agent will be set automatically as its owner).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'RequiredLock'} =  '0';
```

Ticket::Frontend::AgentTicketOwner###TicketType
Sets the ticket type in the ticket owner screen of a zoomed ticket in the agent interface (Ticket::Type needs to be activated).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'TicketType'} =  '0';
```

Ticket::Frontend::AgentTicketOwner###Service
Sets the service in the ticket owner screen of a zoomed ticket in the agent interface (Ticket::Service needs to be activated).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'Service'} =  '0';
```

Ticket::Frontend::AgentTicketOwner###ServiceMandatory
Sets if service must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'ServiceMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketOwner###SLAMandatory
Sets if SLA must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'SLAMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketOwner###Queue
Sets the queue in the ticket owner screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'Queue'} =  '0';
```

Ticket::Frontend::AgentTicketOwner###Owner
Sets the ticket owner in the ticket owner screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'Owner'} =  '1';
```

Ticket::Frontend::AgentTicketOwner###OwnerMandatory
Sets if ticket owner must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'OwnerMandatory'} =  '1';
```

Ticket::Frontend::AgentTicketOwner###Responsible
Sets the responsible agent of the ticket in the ticket owner screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'Responsible'} =  '0';
```

Ticket::Frontend::AgentTicketOwner###State

If a note is added by an agent, sets the state of the ticket in the ticket owner screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'State'} =  '0';
```

Ticket::Frontend::AgentTicketOwner###StateType

Defines the next state of a ticket after adding a note, in the ticket owner screen of a zoomed ticket in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'StateType'} =  [
  'open',
  'pending reminder',
  'pending auto'
];
```

Ticket::Frontend::AgentTicketOwner###StateDefault

Defines the default next state of a ticket after adding a note, in the ticket owner screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'StateDefault'} =  'open';
```

Ticket::Frontend::AgentTicketOwner###Note

Allows adding notes in the ticket owner screen of a zoomed ticket in the agent interface. Can be overwritten by Ticket::Frontend::NeedAccountedTime.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'Note'} =  '1';
```

Ticket::Frontend::AgentTicketOwner###NoteMandatory

Sets if note must be filled in by the agent. Can be overwritten by Ticket::Frontend::NeedAccountedTime.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'NoteMandatory'} =  '1';
```

Ticket::Frontend::AgentTicketOwner###Subject

Sets the default subject for notes added in the ticket owner screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'Subject'} =  '';
```

Ticket::Frontend::AgentTicketOwner###Body

Sets the default body text for notes added in the ticket owner screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'Body'} =  '';
```

Ticket::Frontend::AgentTicketOwner###InvolvedAgent
Shows a list of all the involved agents on this ticket, in the ticket owner screen of a zoomed ticket in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'InvolvedAgent'} =  '0';
```

Ticket::Frontend::AgentTicketOwner###InformAgent
Shows a list of all the possible agents (all agents with note permissions on the queue/ticket) to determine who should be informed about this note, in the ticket owner screen of a zoomed ticket in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'InformAgent'} =  '0';
```

Ticket::Frontend::AgentTicketOwner###ArticleTypeDefault
Defines the default type of the note in the ticket owner screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'ArticleTypeDefault'} =  'note-internal';
```

Ticket::Frontend::AgentTicketOwner###ArticleTypes
Specifies the available note types for this ticket mask. If the option is deselected, ArticleTypeDefault is used and the option is removed from the mask.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'ArticleTypes'} =  {
  'note-external' => '0',
  'note-internal' => '1',
  'note-report' => '0'
};
```

Ticket::Frontend::AgentTicketOwner###Priority
Shows the ticket priority options in the ticket owner screen of a zoomed ticket in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'Priority'} =  '0';
```

Ticket::Frontend::AgentTicketOwner###PriorityDefault
Defines the default ticket priority in the ticket owner screen of a zoomed ticket in the agent interface.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'PriorityDefault'} =  '3 normal';
```

Ticket::Frontend::AgentTicketOwner###Title
Shows the title fields in the ticket owner screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'Title'} =  '0';
```

Ticket::Frontend::AgentTicketOwner###HistoryType
Defines the history type for the ticket owner screen action, which gets used for ticket history in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'HistoryType'} =  'AddNote';
```

Ticket::Frontend::AgentTicketOwner###HistoryComment
Defines the history comment for the ticket owner screen action, which gets used for ticket history in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'HistoryComment'} =  '%%Owner';
```

Ticket::Frontend::AgentTicketOwner###DynamicField
Dynamic fields shown in the ticket owner screen of the agent interface. Possible settings: 0 = Disabled, 1 = Enabled, 2 = Enabled and required.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'DynamicField'} =  {};
```

Ticket::Frontend::AgentTicketOwner###RichTextWidth
Defines the width for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'RichTextWidth'} =  '620';
```

Ticket::Frontend::AgentTicketOwner###RichTextHeight
Defines the height for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketOwner'}->{'RichTextHeight'} =  '100';
```

Ticket → Frontend::Agent::Ticket::ViewPending

Ticket::Frontend::AgentTicketPending###Permission
Required permissions to use the ticket pending screen of a zoomed ticket in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'Permission'} =  'pending';
```

Ticket::Frontend::AgentTicketPending###RequiredLock
Defines if a ticket lock is required in the ticket pending screen of a zoomed ticket in the agent interface (if the ticket isn't locked yet, the ticket gets locked and the current agent will be set automatically as its owner).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'RequiredLock'} =  '1';
```

Ticket::Frontend::AgentTicketPending###TicketType
Sets the ticket type in the ticket pending screen of a zoomed ticket in the agent interface (Ticket::Type needs to be activated).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'TicketType'} =  '0';
```

Ticket::Frontend::AgentTicketPending###Service
Sets the service in the ticket pending screen of a zoomed ticket in the agent interface (Ticket::Service needs to be activated).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'Service'} =  '0';
```

Ticket::Frontend::AgentTicketPending###ServiceMandatory
Sets if service must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'ServiceMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketPending###SLAMandatory
Sets if SLA must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'SLAMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketPending###Queue
Sets the queue in the ticket pending screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'Queue'} =  '0';
```

Ticket::Frontend::AgentTicketPending###Owner
Sets the ticket owner in the ticket pending screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'Owner'} =  '0';
```

Ticket::Frontend::AgentTicketPending###OwnerMandatory
Sets if ticket owner must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'OwnerMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketPending###Responsible
Sets the responsible agent of the ticket in the ticket pending screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'Responsible'} =  '0';
```

Ticket::Frontend::AgentTicketPending###State
If a note is added by an agent, sets the state of the ticket in the ticket pending screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'State'} =  '1';
```

Ticket::Frontend::AgentTicketPending###StateType
Defines the next state of a ticket after adding a note, in the ticket pending screen of a zoomed ticket in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'StateType'} =  [
  'pending reminder',
  'pending auto'
];
```

Ticket::Frontend::AgentTicketPending###StateDefault
Defines the default next state of a ticket after adding a note, in the ticket pending screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'StateDefault'} =  'pending
reminder';
```

Ticket::Frontend::AgentTicketPending###Note
Allows adding notes in the ticket pending screen of a zoomed ticket in the agent interface. Can be overwritten by Ticket::Frontend::NeedAccountedTime.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'Note'} =  '1';
```

Ticket::Frontend::AgentTicketPending###NoteMandatory
Sets if note must be filled in by the agent. Can be overwritten by Ticket::Frontend::NeedAccountedTime.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'NoteMandatory'} =  '1';
```

Ticket::Frontend::AgentTicketPending###Subject
Sets the default subject for notes added in the ticket pending screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'Subject'} =  '';
```

Ticket::Frontend::AgentTicketPending###Body
Sets the default body text for notes added in the ticket pending screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'Body'} =  '';
```

Ticket::Frontend::AgentTicketPending###InvolvedAgent
Shows a list of all the involved agents on this ticket, in the ticket pending screen of a zoomed ticket in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'InvolvedAgent'} =  '0';
```

Ticket::Frontend::AgentTicketPending###InformAgent

Shows a list of all the possible agents (all agents with note permissions on the queue/ticket) to determine who should be informed about this note, in the ticket pending screen of a zoomed ticket in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'InformAgent'} =  '0';
```

Ticket::Frontend::AgentTicketPending###ArticleTypeDefault

Defines the default type of the note in the ticket pending screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'ArticleTypeDefault'} =  'note-internal';
```

Ticket::Frontend::AgentTicketPending###ArticleTypes

Specifies the available note types for this ticket mask. If the option is deselected, ArticleTypeDefault is used and the option is removed from the mask.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'ArticleTypes'} =  {
  'note-external' => '0',
  'note-internal' => '1',
  'note-report' => '0'
};
```

Ticket::Frontend::AgentTicketPending###Priority

Shows the ticket priority options in the ticket pending screen of a zoomed ticket in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'Priority'} =  '0';
```

Ticket::Frontend::AgentTicketPending###PriorityDefault

Defines the default ticket priority in the ticket pending screen of a zoomed ticket in the agent interface.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'PriorityDefault'} =  '3 normal';
```

Ticket::Frontend::AgentTicketPending###Title

Shows the title fields in the ticket pending screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'Title'} =  '0';
```

Ticket::Frontend::AgentTicketPending###HistoryType

Defines the history type for the ticket pending screen action, which gets used for ticket history in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'HistoryType'} =  'AddNote';
```

Ticket::Frontend::AgentTicketPending###HistoryComment
Defines the history comment for the ticket pending screen action, which gets used for ticket history in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'HistoryComment'} =  '%%Pending';
```

Ticket::Frontend::AgentTicketPending###DynamicField
Dynamic fields shown in the ticket pending screen of the agent interface. Possible settings: 0 = Disabled, 1 = Enabled, 2 = Enabled and required.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'DynamicField'} =  {};
```

Ticket::Frontend::AgentTicketPending###RichTextWidth
Defines the width for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'RichTextWidth'} =  '620';
```

Ticket::Frontend::AgentTicketPending###RichTextHeight
Defines the height for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPending'}->{'RichTextHeight'} =  '100';
```

Ticket → Frontend::Agent::Ticket::ViewPhoneInbound

Ticket::Frontend::AgentTicketPhoneInbound###Permission
Required permissions to use the ticket phone inbound screen in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneInbound'}->{'Permission'} =  'phone';
```

Ticket::Frontend::AgentTicketPhoneInbound###RequiredLock
Defines if a ticket lock is required in the ticket phone inbound screen of the agent interface (if the ticket isn't locked yet, the ticket gets locked and the current agent will be set automatically as its owner).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneInbound'}->{'RequiredLock'} =  '0';
```

Ticket::Frontend::AgentTicketPhoneInbound###ArticleType
Defines the default type of the note in the ticket phone inbound screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneInbound'}->{'ArticleType'} =  'phone';
```

Ticket::Frontend::AgentTicketPhoneInbound###SenderType
Defines the default sender type for phone tickets in the ticket phone inbound screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneInbound'}->{'SenderType'} =  'customer';
```

Ticket::Frontend::AgentTicketPhoneInbound###Subject
Defines the default subject for phone tickets in the ticket phone inbound screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneInbound'}->{'Subject'} =  '';
```

Ticket::Frontend::AgentTicketPhoneInbound###Body
Defines the default note body text for phone tickets in the ticket phone inbound screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneInbound'}->{'Body'} =  '';
```

Ticket::Frontend::AgentTicketPhoneInbound###State
Defines the default ticket next state after adding a phone note in the ticket phone inbound screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneInbound'}->{'State'} =  'open';
```

Ticket::Frontend::AgentTicketPhoneInbound###StateType
Next possible ticket states after adding a phone note in the ticket phone inbound screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneInbound'}->{'StateType'} =  [
  'open',
  'pending auto',
  'pending reminder',
  'closed'
];
```

Ticket::Frontend::AgentTicketPhoneInbound###HistoryType
Defines the history type for the ticket phone inbound screen action, which gets used for ticket history in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneInbound'}->{'HistoryType'} =
'PhoneCallCustomer';
```

Ticket::Frontend::AgentTicketPhoneInbound###HistoryComment
Defines the history comment for the ticket phone inbound screen action, which gets used for ticket history in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneInbound'}->{'HistoryComment'} =  '';
```

Ticket::Frontend::AgentTicketPhoneInbound###DynamicField
Dynamic fields shown in the ticket phone inbound screen of the agent interface. Possible settings: 0 = Disabled, 1 = Enabled, 2 = Enabled and required.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneInbound'}->{'DynamicField'} =  {};
```

Ticket::Frontend::AgentTicketPhoneInbound###RichTextWidth
Defines the width for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneInbound'}->{'RichTextWidth'} =  '475';
```

Ticket::Frontend::AgentTicketPhoneInbound###RichTextHeight
Defines the height for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneInbound'}->{'RichTextHeight'} =  '200';
```

Ticket → Frontend::Agent::Ticket::ViewPhoneNew

Ticket::Frontend::AgentTicketPhone###Priority
Sets the default priority for new phone tickets in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhone'}->{'Priority'} =  '3 normal';
```

Ticket::Frontend::AgentTicketPhone###ArticleType
Sets the default article type for new phone tickets in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhone'}->{'ArticleType'} =  'phone';
```

Ticket::Frontend::AgentTicketPhone###SenderType
Sets the default sender type for new phone ticket in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhone'}->{'SenderType'} =  'customer';
```

Ticket::Frontend::AgentTicketPhone::CustomerIDReadOnly
Controls if CutomerID is editable in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhone::CustomerIDReadOnly'} =  '1';
```

Ticket::Frontend::AgentTicketPhone::AllowMultipleFrom
Controls if more than one from entry can be set in the new phone ticket in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhone::AllowMultipleFrom'} =  '1';
```

Ticket::Frontend::AgentTicketPhone###Subject
Sets the default subject for new phone tickets (e.g. 'Phone call') in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhone'}->{'Subject'} =  '';
```

Ticket::Frontend::AgentTicketPhone###Body
Sets the default note text for new telephone tickets. E.g 'New ticket via call' in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhone'}->{'Body'} =  '';
```

Ticket::Frontend::AgentTicketPhone###StateDefault
Sets the default next state for new phone tickets in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhone'}->{'StateDefault'} =  'open';
```

Ticket::Frontend::AgentTicketPhone###StateType
Determines the next possible ticket states, after the creation of a new phone ticket in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhone'}->{'StateType'} =  [
  'open',
  'pending auto',
  'pending reminder',
  'closed'
];
```

Ticket::Frontend::AgentTicketPhone###HistoryType
Defines the history type for the phone ticket screen action, which gets used for ticket history in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhone'}->{'HistoryType'} =  'PhoneCallCustomer';
```

Ticket::Frontend::AgentTicketPhone###HistoryComment
Defines the history comment for the phone ticket screen action, which gets used for ticket history in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhone'}->{'HistoryComment'} =  '';
```

Ticket::Frontend::AgentTicketPhone###SplitLinkType
Sets the default link type of splitted tickets in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhone'}->{'SplitLinkType'} =  {
 'Direction' => 'Target',
 'LinkType' => 'ParentChild'
};
```

Ticket::Frontend::AgentTicketPhone###ServiceMandatory
Sets if service must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhone'}->{'ServiceMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketPhone###SLAMandatory
Sets if SLA must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhone'}->{'SLAMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketPhone###DynamicField
Dynamic fields shown in the ticket phone screen of the agent interface. Possible settings: 0 = Disabled, 1 = Enabled, 2 = Enabled and required.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhone'}->{'DynamicField'} =  {};
```

Ticket::Frontend::AgentTicketPhone###RichTextWidth
Defines the width for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhone'}->{'RichTextWidth'} =  '620';
```

Ticket::Frontend::AgentTicketPhone###RichTextHeight
Defines the height for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhone'}->{'RichTextHeight'} =  '320';
```

Ticket → Frontend::Agent::Ticket::ViewPhoneOutbound

Ticket::Frontend::AgentTicketPhoneOutbound###Permission
Required permissions to use the ticket phone outbound screen in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneOutbound'}->{'Permission'} =  'phone';
```

Ticket::Frontend::AgentTicketPhoneOutbound###RequiredLock
Defines if a ticket lock is required in the ticket phone outbound screen of the agent interface (if the ticket isn't locked yet, the ticket gets locked and the current agent will be set automatically as its owner).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneOutbound'}->{'RequiredLock'} =  '1';
```

Ticket::Frontend::AgentTicketPhoneOutbound###ArticleType
Defines the default type of the note in the ticket phone outbound screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneOutbound'}->{'ArticleType'} =  'phone';
```

Ticket::Frontend::AgentTicketPhoneOutbound###SenderType
Defines the default sender type for phone tickets in the ticket phone outbound screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneOutbound'}->{'SenderType'} =  'agent';
```

Ticket::Frontend::AgentTicketPhoneOutbound###Subject
Defines the default subject for phone tickets in the ticket phone outbound screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneOutbound'}->{'Subject'} =  '';
```

Ticket::Frontend::AgentTicketPhoneOutbound###Body
Defines the default note body text for phone tickets in the ticket phone outbound screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneOutbound'}->{'Body'} =  '';
```

Ticket::Frontend::AgentTicketPhoneOutbound###State
Defines the default ticket next state after adding a phone note in the ticket phone outbound screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneOutbound'}->{'State'} =  'closed
successful';
```

Ticket::Frontend::AgentTicketPhoneOutbound###StateType
Next possible ticket states after adding a phone note in the ticket phone outbound screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneOutbound'}->{'StateType'} =  [
  'open',
  'pending auto',
  'pending reminder',
  'closed'
];
```

Ticket::Frontend::AgentTicketPhoneOutbound###HistoryType
Defines the history type for the ticket phone outbound screen action, which gets used for ticket history in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneOutbound'}->{'HistoryType'} =
'PhoneCallAgent';
```

Ticket::Frontend::AgentTicketPhoneOutbound###HistoryComment
Defines the history comment for the ticket phone outbound screen action, which gets used for ticket history in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneOutbound'}->{'HistoryComment'} =  '';
```

Ticket::Frontend::AgentTicketPhoneOutbound###DynamicField
Dynamic fields shown in the ticket phone outbound screen of the agent interface. Possible settings: 0 = Disabled, 1 = Enabled, 2 = Enabled and required.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneOutbound'}->{'DynamicField'} =  {};
```

Ticket::Frontend::AgentTicketPhoneOutbound###RichTextWidth
Defines the width for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneOutbound'}->{'RichTextWidth'} =  '475';
```

Ticket::Frontend::AgentTicketPhoneOutbound###RichTextHeight
Defines the height for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPhoneOutbound'}->{'RichTextHeight'} =  '200';
```

Ticket → Frontend::Agent::Ticket::ViewPrint

Ticket::Frontend::AgentTicketPrint###DynamicField
Dynamic fields shown in the ticket print screen of the agent interface. Possible settings: 0 = Disabled, 1 = Enabled.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPrint'}->{'DynamicField'} =  {};
```

Ticket → Frontend::Agent::Ticket::ViewPriority

Ticket::Frontend::AgentTicketPriority###Permission
Required permissions to use the ticket priority screen of a zoomed ticket in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'Permission'} =  'priority';
```

Ticket::Frontend::AgentTicketPriority###RequiredLock
Defines if a ticket lock is required in the ticket priority screen of a zoomed ticket in the agent interface (if the ticket isn't locked yet, the ticket gets locked and the current agent will be set automatically as its owner).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'RequiredLock'} =  '1';
```

Ticket::Frontend::AgentTicketPriority###TicketType
Sets the ticket type in the ticket priority screen of a zoomed ticket in the agent interface (Ticket::Type needs to be activated).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'TicketType'} =  '0';
```

Ticket::Frontend::AgentTicketPriority###Service
Sets the service in the ticket priority screen of a zoomed ticket in the agent interface (Ticket::Service needs to be activated).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'Service'} =  '0';
```

Ticket::Frontend::AgentTicketPriority###ServiceMandatory
Sets if service must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'ServiceMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketPriority###SLAMandatory
Sets if SLA must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'SLAMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketPriority###Queue
Sets the queue in the ticket priority screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'Queue'} =  '0';
```

Ticket::Frontend::AgentTicketPriority###Owner
Sets the ticket owner in the ticket priority screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'Owner'} =  '0';
```

Ticket::Frontend::AgentTicketPriority###OwnerMandatory
Sets if ticket owner must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'OwnerMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketPriority###Responsible
Sets the responsible agent of the ticket in the ticket priority screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'Responsible'} =  '0';
```

Ticket::Frontend::AgentTicketPriority###State
If a note is added by an agent, sets the state of the ticket in the ticket priority screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'State'} =  '0';
```

Ticket::Frontend::AgentTicketPriority###StateType
Defines the next state of a ticket after adding a note, in the ticket priority screen of a zoomed ticket in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'StateType'} =  [
  'open',
  'pending reminder',
  'pending auto'
];
```

Ticket::Frontend::AgentTicketPriority###StateDefault
Defines the default next state of a ticket after adding a note, in the ticket priority screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'StateDefault'} =  'open';
```

Ticket::Frontend::AgentTicketPriority###Note
Allows adding notes in the ticket priority screen of a zoomed ticket in the agent interface. Can be overwritten by Ticket::Frontend::NeedAccountedTime.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'Note'} =  '1';
```

Ticket::Frontend::AgentTicketPriority###NoteMandatory
Sets if note must be filled in by the agent. Can be overwritten by Ticket::Frontend::NeedAccountedTime.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'NoteMandatory'} =  '1';
```

Ticket::Frontend::AgentTicketPriority###Subject
Sets the default subject for notes added in the ticket priority screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'Subject'} =  '';
```

Ticket::Frontend::AgentTicketPriority###Body
Sets the default body text for notes added in the ticket priority screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'Body'} =  '';
```

Ticket::Frontend::AgentTicketPriority###InvolvedAgent
Shows a list of all the involved agents on this ticket, in the ticket priority screen of a zoomed ticket in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'InvolvedAgent'} =  '0';
```

Ticket::Frontend::AgentTicketPriority###InformAgent
Shows a list of all the possible agents (all agents with note permissions on the queue/ticket) to determine who should be informed about this note, in the ticket priority screen of a zoomed ticket in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'InformAgent'} =  '0';
```

Ticket::Frontend::AgentTicketPriority###ArticleTypeDefault
Defines the default type of the note in the ticket priority screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'ArticleTypeDefault'} =  'note-internal';
```

Ticket::Frontend::AgentTicketPriority###ArticleTypes
Specifies the available note types for this ticket mask. If the option is deselected, ArticleTypeDefault is used and the option is removed from the mask.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'ArticleTypes'} =  {
  'note-external' => '0',
  'note-internal' => '1',
  'note-report' => '0'
};
```

Ticket::Frontend::AgentTicketPriority###Priority
Shows the ticket priority options in the ticket priority screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'Priority'} =  '1';
```

Ticket::Frontend::AgentTicketPriority###PriorityDefault
Defines the default ticket priority in the ticket priority screen of a zoomed ticket in the agent interface.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'PriorityDefault'} =  '3 normal';
```

Ticket::Frontend::AgentTicketPriority###Title
Shows the title fields in the ticket priority screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'Title'} =  '0';
```

Ticket::Frontend::AgentTicketPriority###HistoryType
Defines the history type for the ticket priority screen action, which gets used for ticket history in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'HistoryType'} =  'AddNote';
```

Ticket::Frontend::AgentTicketPriority###HistoryComment
Defines the history comment for the ticket priority screen action, which gets used for ticket history in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'HistoryComment'} =  '%%Priority';
```

Ticket::Frontend::AgentTicketPriority###DynamicField
Dynamic fields shown in the ticket priority screen of the agent interface. Possible settings: 0 = Disabled, 1 = Enabled, 2 = Enabled and required.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'DynamicField'} =  {};
```

Ticket::Frontend::AgentTicketPriority###RichTextWidth
Defines the width for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'RichTextWidth'} =  '620';
```

Ticket::Frontend::AgentTicketPriority###RichTextHeight
Defines the height for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketPriority'}->{'RichTextHeight'} =  '100';
```

Ticket → Frontend::Agent::Ticket::ViewQueue

Ticket::Frontend::AgentTicketQueue###StripEmptyLines
Strips empty lines on the ticket preview in the queue view.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketQueue'}->{'StripEmptyLines'} =  '0';
```

Ticket::Frontend::AgentTicketQueue###ViewAllPossibleTickets
Shows all both ro and rw queues in the queue view.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketQueue'}->{'ViewAllPossibleTickets'} =  '0';
```

Ticket::Frontend::AgentTicketQueue###HideEmptyQueues
Show queues even when only locked tickets are in.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketQueue'}->{'HideEmptyQueues'} =  '0';
```

Ticket::Frontend::AgentTicketQueue###HighlightAge1
Sets the age in minutes (first level) for highlighting queues that contain untouched tickets.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketQueue'}->{'HighlightAge1'} =  '1440';
```

Ticket::Frontend::AgentTicketQueue###HighlightAge2
Sets the age in minutes (second level) for highlighting queues that contain untouched tickets.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketQueue'}->{'HighlightAge2'} =  '2880';
```

Ticket::Frontend::AgentTicketQueue###Blink
Activates a blinking mechanism of the queue that contains the oldest ticket.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketQueue'}->{'Blink'} =  '1';
```

Ticket::Frontend::AgentTicketQueue###UseSubQueues
Include tickets of subqueues per default when selecting a queue.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketQueue'}->{'UseSubQueues'} =  '0';
```

Ticket::Frontend::AgentTicketQueue###QueueSort
Sorts the tickets (ascendingly or descendingly) when a single queue is selected in the queue view and after the tickets are sorted by priority. Values: 0 = ascending (oldest on top, default), 1 = descending (youngest on top). Use the QueueID for the key and 0 or 1 for value.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketQueue'}->{'QueueSort'} =  {
  '3' => '0',
  '7' => '1'
};
```

Ticket::Frontend::AgentTicketQueue###SortBy::Default
Defines the default sort criteria for all queues displayed in the queue view.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketQueue'}->{'SortBy::Default'} =  'Age';
```

Ticket::Frontend::AgentTicketQueue###PreSort::ByPriority
Defines if a pre-sorting by priority should be done in the queue view.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketQueue'}->{'PreSort::ByPriority'} =  '1';
```

Ticket::Frontend::AgentTicketQueue###Order::Default
Defines the default sort order for all queues in the queue view, after priority sort.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketQueue'}->{'Order::Default'} =  'Up';
```

Ticket::Frontend::AgentTicketQueue###DefaultColumns
Columns that can be filtered in the queue view of the agent interface. Possible settings: 0 = Disabled, 1 = Available, 2 = Enabled by default. Note: Only Ticket attributes, Dynamic Fields (DynamicField_NameX) and Customer attributes (e.g. CustomerUserPhone, CustomerCompanyName, ...) are allowed.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketQueue'}->{'DefaultColumns'} =  {
  'Age' => '2',
  'Changed' => '1',
  'Created' => '1',
  'CustomerCompanyName' => '1',
  'CustomerID' => '2',
  'CustomerName' => '1',
  'CustomerUserID' => '1',
  'EscalationResponseTime' => '1',
  'EscalationSolutionTime' => '1',
  'EscalationTime' => '1',
  'EscalationUpdateTime' => '1',
  'Lock' => '2',
  'Owner' => '2',
  'PendingTime' => '1',
  'Priority' => '1',
  'Queue' => '2',
  'Responsible' => '1',
  'SLA' => '1',
  'Service' => '1',
  'State' => '2',
  'TicketNumber' => '2',
  'Title' => '2',
  'Type' => '1'
```

```
};
```

Ticket → Frontend::Agent::Ticket::ViewResponsible

Ticket::Frontend::AgentTicketResponsibleView###SortBy::Default
Defines the default ticket attribute for ticket sorting in the responsible view of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsibleView'}->{'SortBy::Default'} =  'Age';
```

Ticket::Frontend::AgentTicketResponsibleView###Order::Default
Defines the default ticket order in the responsible view of the agent interface. Up: oldest on top. Down: latest on top.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsibleView'}->{'Order::Default'} =  'Up';
```

Ticket::Frontend::AgentTicketResponsible###Permission
Required permissions to use the ticket responsible screen in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'Permission'} =  'responsible';
```

Ticket::Frontend::AgentTicketResponsible###RequiredLock
Defines if a ticket lock is required in the ticket responsible screen of the agent interface (if the ticket isn't locked yet, the ticket gets locked and the current agent will be set automatically as its owner).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'RequiredLock'} =  '0';
```

Ticket::Frontend::AgentTicketResponsible###TicketType
Sets the ticket type in the ticket responsible screen of the agent interface (Ticket::Type needs to be activated).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'TicketType'} =  '0';
```

Ticket::Frontend::AgentTicketResponsible###Service
Sets the service in the ticket responsible screen of the agent interface (Ticket::Service needs to be activated).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'Service'} =  '0';
```

Ticket::Frontend::AgentTicketResponsible###ServiceMandatory
Sets if service must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'ServiceMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketResponsible###SLAMandatory
Sets if SLA must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'SLAMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketResponsible###Queue
Sets the queue in the ticket responsible screen of a zoomed ticket in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'Queue'} =  '0';
```

Ticket::Frontend::AgentTicketResponsible###Owner
Sets the ticket owner in the ticket responsible screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'Owner'} =  '0';
```

Ticket::Frontend::AgentTicketResponsible###OwnerMandatory
Sets if ticket owner must be selected by the agent.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'OwnerMandatory'} =  '0';
```

Ticket::Frontend::AgentTicketResponsible###Responsible
Sets the responsible agent of the ticket in the ticket responsible screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'Responsible'} =  '1';
```

Ticket::Frontend::AgentTicketResponsible###State
If a note is added by an agent, sets the state of a ticket in the ticket responsible screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'State'} =  '0';
```

Ticket::Frontend::AgentTicketResponsible###StateType
Defines the next state of a ticket after adding a note, in the ticket responsible screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'StateType'} =  [
  'open',
  'pending reminder',
  'pending auto'
];
```

Ticket::Frontend::AgentTicketResponsible###StateDefault
Defines the default next state of a ticket after adding a note, in the ticket responsible screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'StateDefault'} =  'open';
```

Ticket::Frontend::AgentTicketResponsible###Note

Allows adding notes in the ticket responsible screen of the agent interface. Can be overwritten by Ticket::Frontend::NeedAccountedTime.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'Note'} =  '1';
```

Ticket::Frontend::AgentTicketResponsible###NoteMandatory

Sets if note must be filled in by the agent. Can be overwritten by Ticket::Frontend::NeedAccountedTime.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'NoteMandatory'} =  '1';
```

Ticket::Frontend::AgentTicketResponsible###Subject

Sets the default subject for notes added in the ticket responsible screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'Subject'} =  '';
```

Ticket::Frontend::AgentTicketResponsible###Body

Sets the default body text for notes added in the ticket responsible screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'Body'} =  '';
```

Ticket::Frontend::AgentTicketResponsible###InvolvedAgent

Shows a list of all the involved agents on this ticket, in the ticket responsible screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'InvolvedAgent'} =  '0';
```

Ticket::Frontend::AgentTicketResponsible###InformAgent

Shows a list of all the possible agents (all agents with note permissions on the queue/ticket) to determine who should be informed about this note, in the ticket responsible screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'InformAgent'} =  '0';
```

Ticket::Frontend::AgentTicketResponsible###ArticleTypeDefault

Defines the default type of the note in the ticket responsible screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'ArticleTypeDefault'} =  'note-internal';
```

Ticket::Frontend::AgentTicketResponsible###ArticleTypes

Specifies the available note types for this ticket mask. If the option is deselected, ArticleTypeDefault is used and the option is removed from the mask.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'ArticleTypes'} =  {
  'note-external' => '0',
  'note-internal' => '1',
  'note-report' => '0'
};
```

Ticket::Frontend::AgentTicketResponsible###Priority
Shows the ticket priority options in the ticket responsible screen of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'Priority'} =  '0';
```

Ticket::Frontend::AgentTicketResponsible###PriorityDefault
Defines the default ticket priority in the ticket responsible screen of the agent interface.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'PriorityDefault'} =  '3 normal';
```

Ticket::Frontend::AgentTicketResponsible###Title
Shows the title fields in the ticket responsible screen of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'Title'} =  '1';
```

Ticket::Frontend::AgentTicketResponsible###HistoryType
Defines the history type for the ticket responsible screen action, which gets used for ticket history in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'HistoryType'} =  'AddNote';
```

Ticket::Frontend::AgentTicketResponsible###HistoryComment
Defines the history comment for the ticket responsible screen action, which gets used for ticket history in the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'HistoryComment'} =  '%
%Responsible';
```

Ticket::Frontend::AgentTicketResponsible###DynamicField
Dynamic fields shown in the ticket responsible screen of the agent interface. Possible settings: 0 = Disabled, 1 = Enabled, 2 = Enabled and required.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'DynamicField'} =  {};
```

Ticket::Frontend::AgentTicketResponsible###RichTextWidth
Defines the width for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'RichTextWidth'} =  '620';
```

Ticket::Frontend::AgentTicketResponsible###RichTextHeight

Defines the height for the rich text editor component for this screen. Enter number (pixels) or percent value (relative).

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsible'}->{'RichTextHeight'} =  '100';
```

Ticket::Frontend::AgentTicketResponsibleView###DefaultColumns

Columns that can be filtered in the responsible view of the agent interface. Possible settings: 0 = Disabled, 1 = Available, 2 = Enabled by default. Note: Only Ticket attributes, Dynamic Fields (DynamicField_NameX) and Customer attributes (e.g. CustomerUserPhone, CustomerCompanyName, ...) are allowed.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketResponsibleView'}->{'DefaultColumns'} =  {
  'Age' => '2',
  'Changed' => '1',
  'Created' => '1',
  'CustomerCompanyName' => '1',
  'CustomerID' => '2',
  'CustomerName' => '1',
  'CustomerUserID' => '1',
  'EscalationResponseTime' => '1',
  'EscalationSolutionTime' => '1',
  'EscalationTime' => '1',
  'EscalationUpdateTime' => '1',
  'Lock' => '2',
  'Owner' => '2',
  'PendingTime' => '1',
  'Priority' => '1',
  'Queue' => '2',
  'Responsible' => '1',
  'SLA' => '1',
  'Service' => '1',
  'State' => '2',
  'TicketNumber' => '2',
  'Title' => '2',
  'Type' => '1'
};
```

Ticket → Frontend::Agent::Ticket::ViewSearch

Ticket::Frontend::AgentTicketSearch###ExtendedSearchCondition

Allows extended search conditions in ticket search of the agent interface. With this feature you can search e. g. with this kind of conditions like "(key1&&key2)" or "(key1||key2)".

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'ExtendedSearchCondition'} =  '1';
```

Ticket::Frontend::AgentTicketSearch###SearchLimit

Maximum number of tickets to be displayed in the result of a search in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'SearchLimit'} =  '2000';
```

Ticket::Frontend::AgentTicketSearch###SearchPageShown
Number of tickets to be displayed in each page of a search result in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'SearchPageShown'} =  '40';
```

Ticket::Frontend::AgentTicketSearch###SearchViewableTicketLines
Number of lines (per ticket) that are shown by the search utility in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'SearchViewableTicketLines'} =  '10';
```

Ticket::Frontend::AgentTicketSearch###SortBy::Default
Defines the default ticket attribute for ticket sorting of the ticket search result of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'SortBy::Default'} =  'Age';
```

Ticket::Frontend::AgentTicketSearch###Order::Default
Defines the default ticket order in the ticket search result of the agent interface. Up: oldest on top. Down: latest on top.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Order::Default'} =  'Down';
```

Ticket::Frontend::AgentTicketSearch###SearchArticleCSVTree
Exports the whole article tree in search result (it can affect the system performance).

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'SearchArticleCSVTree'} =  '0';
```

Ticket::Frontend::AgentTicketSearch###SearchCSVData
Data used to export the search result in CSV format.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'SearchCSVData'} =  [
 'TicketNumber',
 'Age',
 'Created',
 'Closed',
 'FirstLock',
 'FirstResponse',
 'State',
 'Priority',
 'Queue',
 'Lock',
 'Owner',
```

```
  'UserFirstname',
  'UserLastname',
  'CustomerID',
  'CustomerName',
  'From',
  'Subject',
  'AccountedTime',
  'ArticleTree',
  'SolutionInMin',
  'SolutionDiffInMin',
  'FirstResponseInMin',
  'FirstResponseDiffInMin'
];
```

Ticket::Frontend::AgentTicketSearch###ArticleCreateTime
Includes article create times in the ticket search of the agent interface.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'ArticleCreateTime'} =  '0';
```

Ticket::Frontend::AgentTicketSearch###Defaults###Fulltext
Defines the default shown ticket search attribute for ticket search screen.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}->{'Fulltext'} =  '';
```

Ticket::Frontend::AgentTicketSearch###Defaults###TicketNumber
Defines the default shown ticket search attribute for ticket search screen.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}->{'TicketNumber'} =  '';
```

Ticket::Frontend::AgentTicketSearch###Defaults###Title
Defines the default shown ticket search attribute for ticket search screen.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}->{'Title'} =  '';
```

Ticket::Frontend::AgentTicketSearch###Defaults###From
Defines the default shown ticket search attribute for ticket search screen.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}->{'From'} =  '';
```

Ticket::Frontend::AgentTicketSearch###Defaults###To
Defines the default shown ticket search attribute for ticket search screen.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}->{'To'} =  '';
```

Ticket::Frontend::AgentTicketSearch###Defaults###Cc
Defines the default shown ticket search attribute for ticket search screen.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}->{'Cc'} =  '';
```

Ticket::Frontend::AgentTicketSearch###Defaults###Subject
Defines the default shown ticket search attribute for ticket search screen.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}->{'Subject'} =  '';
```

Ticket::Frontend::AgentTicketSearch###Defaults###Body
Defines the default shown ticket search attribute for ticket search screen.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}->{'Body'} =  '';
```

Ticket::Frontend::AgentTicketSearch###Defaults###CustomerID
Defines the default shown ticket search attribute for ticket search screen.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}->{'CustomerID'} =  '';
```

Ticket::Frontend::AgentTicketSearch###Defaults###CustomerUserLogin
Defines the default shown ticket search attribute for ticket search screen.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}->{'CustomerUserLogin'} =
'';
```

Ticket::Frontend::AgentTicketSearch###Defaults###StateIDs
Defines the default shown ticket search attribute for ticket search screen.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}->{'StateIDs'} =  [];
```

Ticket::Frontend::AgentTicketSearch###Defaults###QueueIDs
Defines the default shown ticket search attribute for ticket search screen.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}->{'QueueIDs'} =  [];
```

Ticket::Frontend::AgentTicketSearch###Defaults###TicketCreateTimePoint
Default data to use on attribute for ticket search screen. Example: "TicketCreateTimePointFormat=year;TicketCreateTimePointStart=Last;TicketCreateTimePoint=2;".

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}->{'TicketCreateTimePoint'}
=  '';
```

Ticket::Frontend::AgentTicketSearch###Defaults###TicketCreateTimeSlot

Default data to use on attribute for ticket search screen. Example: "TicketCreateTimeStartYear=2010;TicketCreateTimeStartMonth=10;TicketCreateTimeStartDay=4;T

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}->{'TicketCreateTimeSlot'}
=  '';
```

Ticket::Frontend::AgentTicketSearch###Defaults###TicketChangeTimePoint

Defines the default shown ticket search attribute for ticket search screen.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}->{'TicketChangeTimePoint'}
=  '';
```

Ticket::Frontend::AgentTicketSearch###Defaults###TicketChangeTimeSlot

Defines the default shown ticket search attribute for ticket search screen.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}->{'TicketChangeTimeSlot'}
=  '';
```

Ticket::Frontend::AgentTicketSearch###Defaults###TicketCloseTimePoint

Defines the default shown ticket search attribute for ticket search screen.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}->{'TicketCloseTimePoint'}
=  '';
```

Ticket::Frontend::AgentTicketSearch###Defaults###TicketCloseTimeSlot

Defines the default shown ticket search attribute for ticket search screen.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}->{'TicketCloseTimeSlot'} =
 '';
```

Ticket::Frontend::AgentTicketSearch###Defaults###TicketEscalationTimePoint

Defines the default shown ticket search attribute for ticket search screen.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}-
>{'TicketEscalationTimePoint'} =  '';
```

Ticket::Frontend::AgentTicketSearch###Defaults###TicketEscalationTimeSlot
Defines the default shown ticket search attribute for ticket search screen.

This setting is not active by default.

Default value:

```
 $Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}-
>{'TicketEscalationTimeSlot'} =  '';
```

Ticket::Frontend::AgentTicketSearch###Defaults###ArticleCreateTimePoint
Defines the default shown ticket search attribute for ticket search screen.

This setting is not active by default.

Default value:

```
 $Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}-
>{'ArticleCreateTimePoint'} =  '';
```

Ticket::Frontend::AgentTicketSearch###Defaults###ArticleCreateTimeSlot
Defines the default shown ticket search attribute for ticket search screen.

This setting is not active by default.

Default value:

```
 $Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}->{'ArticleCreateTimeSlot'}
=  '';
```

Ticket::Frontend::AgentTicketSearch###Defaults###SearchInArchive
Defines the default shown ticket search attribute for ticket search screen (AllTickets/ArchivedTickets/NotArchivedTickets).

This setting is not active by default.

Default value:

```
 $Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}->{'SearchInArchive'} =
'';
```

Ticket::Frontend::AgentTicketSearch###DynamicField
Dynamic fields shown in the ticket search screen of the agent interface. Possible settings: 0 = Disabled, 1 = Enabled, 2 = Enabled and shown by default.

Default value:

```
 $Self->{'Ticket::Frontend::AgentTicketSearch'}->{'DynamicField'} =  {};
```

Ticket::Frontend::AgentTicketSearch###Defaults###DynamicField
Defines the default shown ticket search attribute for ticket search screen. Example: "Key" must have the name of the Dynamic Field in this case 'X', "Content" must have the value of the Dynamic Field depending on the Dynamic Field type, Text: 'a text', Dropdown: '1', Date/Time: 'Search_DynamicField_XTimeSlotStartYear=1974;
Search_DynamicField_XTimeSlotStartMonth=01;
Search_DynamicField_XTimeSlotStartDay=26;
Search_DynamicField_XTimeSlotStartHour=00;
Search_DynamicField_XTimeSlotStartMinute=00;
Search_DynamicField_XTimeSlotStartSecond=00;
Search_DynamicField_XTimeSlotStopYear=2013;
Search_DynamicField_XTimeSlotStopMonth=01;
Search_DynamicField_XTimeSlotStopDay=26;
Search_DynamicField_XTimeSlotStopHour=23;

Search_DynamicField_XTimeSlotStopMinute=59;
Search_DynamicField_XTimeSlotStopSecond=59;' and or
'Search_DynamicField_XTimePointFormat=week;
Search_DynamicField_XTimePointStart=Before;
Search_DynamicField_XTimePointValue=7';.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'Defaults'}->{'DynamicField'} =  {};
```

Ticket::Frontend::AgentTicketSearch###SearchCSVDynamicField
Dynamic Fields used to export the search result in CSV format.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'SearchCSVDynamicField'} =  {};
```

Ticket::Frontend::AgentTicketSearch###DefaultColumns
Columns that can be filtered in the ticket search result view of the agent interface. Possible settings: 0 = Disabled, 1 = Available, 2 = Enabled by default. Note: Only Ticket attributes, Dynamic Fields (DynamicField_NameX) and Customer attributes (e.g. CustomerUserPhone, CustomerCompanyName, ...) are allowed.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketSearch'}->{'DefaultColumns'} =  {
  'Age' => '2',
  'Changed' => '1',
  'Created' => '1',
  'CustomerCompanyName' => '1',
  'CustomerID' => '2',
  'CustomerName' => '1',
  'CustomerUserID' => '1',
  'EscalationResponseTime' => '1',
  'EscalationSolutionTime' => '1',
  'EscalationTime' => '1',
  'EscalationUpdateTime' => '1',
  'Lock' => '2',
  'Owner' => '2',
  'PendingTime' => '1',
  'Priority' => '1',
  'Queue' => '2',
  'Responsible' => '1',
  'SLA' => '1',
  'Service' => '1',
  'State' => '2',
  'TicketNumber' => '2',
  'Title' => '2',
  'Type' => '1'
};
```

Ticket → Frontend::Agent::Ticket::ViewService

Ticket::Frontend::AgentTicketService###StripEmptyLines
Strips empty lines on the ticket preview in the service view.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketService'}->{'StripEmptyLines'} =  '0';
```

Ticket::Frontend::AgentTicketService###ViewAllPossibleTickets
Shows all both ro and rw tickets in the service view.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketService'}->{'ViewAllPossibleTickets'} =  '0';
```

Ticket::Frontend::AgentTicketService###ServiceSort

Sorts the tickets (ascendingly or descendingly) when a single queue is selected in the service view and after the tickets are sorted by priority. Values: 0 = ascending (oldest on top, default), 1 = descending (youngest on top). Use the ServiceID for the key and 0 or 1 for value.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketService'}->{'ServiceSort'} =  {
  '3' => '0',
  '7' => '1'
};
```

Ticket::Frontend::AgentTicketService###SortBy::Default

Defines the default sort criteria for all services displayed in the service view.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketService'}->{'SortBy::Default'} =  'Age';
```

Ticket::Frontend::AgentTicketService###PreSort::ByPriority

Defines if a pre-sorting by priority should be done in the service view.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketService'}->{'PreSort::ByPriority'} =  '1';
```

Ticket::Frontend::AgentTicketService###Order::Default

Defines the default sort order for all services in the service view, after priority sort.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketService'}->{'Order::Default'} =  'Up';
```

Ticket::Frontend::AgentTicketService###DefaultColumns

Columns that can be filtered in the service view of the agent interface. Possible settings: 0 = Disabled, 1 = Available, 2 = Enabled by default. Note: Only Ticket attributes, Dynamic Fields (DynamicField_NameX) and Customer attributes (e.g. CustomerUserPhone, CustomerCompanyName, ...) are allowed.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketService'}->{'DefaultColumns'} =  {
  'Age' => '2',
  'Changed' => '1',
  'Created' => '1',
  'CustomerCompanyName' => '1',
  'CustomerID' => '2',
  'CustomerName' => '1',
  'CustomerUserID' => '1',
  'EscalationResponseTime' => '1',
  'EscalationSolutionTime' => '1',
  'EscalationTime' => '1',
  'EscalationUpdateTime' => '1',
```

```
  'Lock' => '2',
  'Owner' => '2',
  'PendingTime' => '1',
  'Priority' => '1',
  'Queue' => '2',
  'Responsible' => '1',
  'SLA' => '1',
  'Service' => '2',
  'State' => '2',
  'TicketNumber' => '2',
  'Title' => '2',
  'Type' => '1'
};
```

Ticket → Frontend::Agent::Ticket::ViewStatus

Ticket::Frontend::AgentTicketStatusView###ViewableTicketsPage
Shows all open tickets (even if they are locked) in the status view of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketStatusView'}->{'ViewableTicketsPage'} =  '50';
```

Ticket::Frontend::AgentTicketStatusView###SortBy::Default
Defines the default ticket attribute for ticket sorting in the status view of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketStatusView'}->{'SortBy::Default'} =  'Age';
```

Ticket::Frontend::AgentTicketStatusView###Order::Default
Defines the default ticket order (after priority sort) in the status view of the agent interface. Up: oldest on top. Down: latest on top.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketStatusView'}->{'Order::Default'} =  'Down';
```

Ticket::Frontend::AgentTicketStatusView###DefaultColumns
Columns that can be filtered in the status view of the agent interface. Possible settings: 0 = Disabled, 1 = Available, 2 = Enabled by default. Note: Only Ticket attributes, Dynamic Fields (DynamicField_NameX) and Customer attributes (e.g. CustomerUserPhone, CustomerCompanyName, ...) are allowed.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketStatusView'}->{'DefaultColumns'} =  {
  'Age' => '2',
  'Changed' => '1',
  'Created' => '1',
  'CustomerCompanyName' => '1',
  'CustomerID' => '2',
  'CustomerName' => '1',
  'CustomerUserID' => '1',
  'EscalationResponseTime' => '1',
  'EscalationSolutionTime' => '1',
  'EscalationTime' => '1',
  'EscalationUpdateTime' => '1',
  'Lock' => '2',
```

```
  'Owner' => '2',
  'PendingTime' => '1',
  'Priority' => '1',
  'Queue' => '2',
  'Responsible' => '1',
  'SLA' => '1',
  'Service' => '1',
  'State' => '2',
  'TicketNumber' => '2',
  'Title' => '2',
  'Type' => '1'
};
```

Ticket → Frontend::Agent::Ticket::ViewWatch

Ticket::Frontend::AgentTicketWatchView###SortBy::Default

Defines the default ticket attribute for ticket sorting in the watch view of the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketWatchView'}->{'SortBy::Default'} =  'Age';
```

Ticket::Frontend::AgentTicketWatchView###Order::Default

Defines the default ticket order in the watch view of the agent interface. Up: oldest on top. Down: latest on top.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketWatchView'}->{'Order::Default'} =  'Up';
```

Ticket::Frontend::AgentTicketWatchView###DefaultColumns

Columns that can be filtered in the watch view of the agent interface. Possible settings: 0 = Disabled, 1 = Available, 2 = Enabled by default. Note: Only Ticket attributes, Dynamic Fields (DynamicField_NameX) and Customer attributes (e.g. CustomerUserPhone, CustomerCompanyName, ...) are allowed.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketWatchView'}->{'DefaultColumns'} =  {
  'Age' => '2',
  'Changed' => '1',
  'Created' => '1',
  'CustomerCompanyName' => '1',
  'CustomerID' => '2',
  'CustomerName' => '1',
  'CustomerUserID' => '1',
  'EscalationResponseTime' => '1',
  'EscalationSolutionTime' => '1',
  'EscalationTime' => '1',
  'EscalationUpdateTime' => '1',
  'Lock' => '2',
  'Owner' => '2',
  'PendingTime' => '1',
  'Priority' => '1',
  'Queue' => '2',
  'Responsible' => '1',
  'SLA' => '1',
  'Service' => '1',
  'State' => '2',
  'TicketNumber' => '2',
  'Title' => '2',
  'Type' => '1'
};
```

Ticket → Frontend::Agent::Ticket::ViewZoom

Ticket::Frontend::PlainView
Shows a link to see a zoomed email ticket in plain text.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::PlainView'} =  '0';
```

Ticket::Frontend::ZoomExpand
Shows all the articles of the ticket (expanded) in the zoom view.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::ZoomExpand'} =  '0';
```

Ticket::Frontend::ZoomExpandSort
Shows the articles sorted normally or in reverse, under ticket zoom in the agent interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::ZoomExpandSort'} =  'reverse';
```

Ticket::ZoomAttachmentDisplayCount
Shows a count of icons in the ticket zoom, if the article has attachments.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::ZoomAttachmentDisplayCount'} =  '20';
```

Ticket::ZoomTimeDisplay
Displays the accounted time for an article in the ticket zoom view.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::ZoomTimeDisplay'} =  '0';
```

Ticket::UseArticleColors
Shows colors for different article types in the article table.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::UseArticleColors'} =  '1';
```

Ticket::Frontend::TicketArticleFilter
Activates the article filter in the zoom view to specify which articles should be shown.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::TicketArticleFilter'} =  '0';
```

Ticket::Frontend::HTMLArticleHeightDefault
Set the default height (in pixels) of inline HTML articles in AgentTicketZoom.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::HTMLArticleHeightDefault'} =  '100';
```

Ticket::Frontend::HTMLArticleHeightMax
Set the maximum height (in pixels) of inline HTML articles in AgentTicketZoom.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::HTMLArticleHeightMax'} =  '2500';
```

Ticket::Frontend::MaxArticlesZoomExpand
The maximal number of articles expanded on a single page in AgentTicketZoom.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::MaxArticlesZoomExpand'} =  '400';
```

Ticket::Frontend::MaxArticlesPerPage
The maximal number of articles shown on a single page in AgentTicketZoom.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::MaxArticlesPerPage'} =  '1000';
```

Ticket::Frontend::ZoomRichTextForce
Show article as rich text even if rich text writing is disabled.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::ZoomRichTextForce'} =  '1';
```

Ticket::Frontend::AgentTicketZoom###DynamicField
Dynamic fields shown in the sidebar of the ticket zoom screen of the agent interface. Possible settings: 0 = Disabled, 1 = Enabled.

Default value:

```
$Self->{'Ticket::Frontend::AgentTicketZoom'}->{'DynamicField'} =  {};
```

Ticket::Frontend::ZoomCollectMeta
Whether or not to collect meta information from articles using filters configured in Ticket::Frontend::ZoomCollectMetaFilters.

Default value:

```
$Self->{'Ticket::Frontend::ZoomCollectMeta'} =  '0';
```

Ticket::Frontend::ZoomCollectMetaFilters###CVE-Mitre
Defines a filter to collect CVE numbers from article texts in AgentTicketZoom. The results will be displayed in a meta box next to the article. Fill in URLPreview if you would like to see a preview when moving your mouse cursor above the link element.

This could be the same URL as in URL, but also an alternate one. Please note that some websites deny being displayed within an iframe (e.g. Google) and thus won't work with the preview mode.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::ZoomCollectMetaFilters'}->{'CVE-Mitre'} =  {
  'Meta' => {
    'Name' => 'CVE Mitre',
    'Target' => '_blank',
    'URL' => 'http://cve.mitre.org/cgi-bin/cvename.cgi?name=<MATCH1>-<MATCH2>-<MATCH3>',
    'URLPreview' => 'http://cve.mitre.org/cgi-bin/cvename.cgi?name=<MATCH1>-<MATCH2>-
<MATCH3>'
  },
  'RegExp' => [
    '(CVE|CAN)\\-(\\d{3,4})\\-(\\d{2,})'
  ]
};
```

Ticket::Frontend::ZoomCollectMetaFilters###CVE-Google

Defines a filter to collect CVE numbers from article texts in AgentTicketZoom. The results will be displayed in a meta box next to the article. Fill in URLPreview if you would like to see a preview when moving your mouse cursor above the link element. This could be the same URL as in URL, but also an alternate one. Please note that some websites deny being displayed within an iframe (e.g. Google) and thus won't work with the preview mode.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::ZoomCollectMetaFilters'}->{'CVE-Google'} =  {
  'Meta' => {
    'Name' => 'CVE Google Search',
    'Target' => '_blank',
    'URL' => 'http://google.com/search?q=<MATCH1>-<MATCH2>-<MATCH3>',
    'URLPreview' => ''
  },
  'RegExp' => [
    '(CVE|CAN)\\-(\\d{3,4})\\-(\\d{2,})'
  ]
};
```

Ticket → Frontend::Agent::TicketOverview

Ticket::Frontend::Overview###Small

Allows having a small format ticket overview (CustomerInfo => 1 - shows also the customer information).

Default value:

```
$Self->{'Ticket::Frontend::Overview'}->{'Small'} =  {
  'CustomerInfo' => '1',
  'Module' => 'Kernel::Output::HTML::TicketOverview::Small',
  'ModulePriority' => '100',
  'Name' => 'Small',
  'NameShort' => 'S'
};
```

Ticket::Frontend::OverviewSmall###ColumnHeader

Shows either the last customer article's subject or the ticket title in the small format overview.

Default value:

```
$Self->{'Ticket::Frontend::OverviewSmall'}->{'ColumnHeader'} =  'LastCustomerSubject';
```

Ticket::Frontend::Overview###Medium

Allows having a medium format ticket overview (CustomerInfo => 1 - shows also the customer information).

Default value:

```
$Self->{'Ticket::Frontend::Overview'}->{'Medium'} =  {
  'CustomerInfo' => '0',
  'Module' => 'Kernel::Output::HTML::TicketOverview::Medium',
  'ModulePriority' => '200',
  'Name' => 'Medium',
  'NameShort' => 'M',
  'OverviewMenuModules' => '1',
  'TicketActionsPerTicket' => '1'
};
```

Ticket::Frontend::Overview###Preview

Shows a preview of the ticket overview (CustomerInfo => 1 - shows also Customer-Info, CustomerInfoMaxSize max. size in characters of Customer-Info).

Default value:

```
$Self->{'Ticket::Frontend::Overview'}->{'Preview'} =  {
  'CustomerInfo' => '0',
  'CustomerInfoMaxSize' => '18',
  'DefaultPreViewLines' => '25',
  'DefaultViewNewLine' => '90',
  'Module' => 'Kernel::Output::HTML::TicketOverview::Preview',
  'ModulePriority' => '300',
  'Name' => 'Large',
  'NameShort' => 'L',
  'OverviewMenuModules' => '1',
  'StripEmptyLines' => '0',
  'TicketActionsPerTicket' => '1'
};
```

Ticket::Frontend::Overview::PreviewArticleSenderTypes

Defines which article sender types should be shown in the preview of a ticket.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::Overview::PreviewArticleSenderTypes'} =  {
  'agent' => '1',
  'customer' => '1',
  'system' => '1'
};
```

Ticket::Frontend::Overview::PreviewArticleLimit

Sets the count of articles visible in preview mode of ticket overviews.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::Overview::PreviewArticleLimit'} =  '5';
```

Ticket::Frontend::Overview::PreviewArticleTypeExpanded

Defines wich article type should be expanded when entering the overview. If nothing defined, latest article will be expanded.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::Overview::PreviewArticleTypeExpanded'} =  '';
```

Ticket::Frontend::OverviewSmall###DynamicField

Dynamic fields shown in the ticket small format overview screen of the agent interface. Possible settings: 0 = Disabled, 1 = Available, 2 = Enabled by default.

Default value:

```
$Self->{'Ticket::Frontend::OverviewSmall'}->{'DynamicField'} =  {};
```

Ticket::Frontend::OverviewMedium###DynamicField

Dynamic fields shown in the ticket medium format overview screen of the agent interface. Possible settings: 0 = Disabled, 1 = Enabled.

Default value:

```
$Self->{'Ticket::Frontend::OverviewMedium'}->{'DynamicField'} =  {};
```

Ticket::Frontend::OverviewPreview###DynamicField

Dynamic fields shown in the ticket preview format overview screen of the agent interface. Possible settings: 0 = Disabled, 1 = Enabled.

Default value:

```
$Self->{'Ticket::Frontend::OverviewPreview'}->{'DynamicField'} =  {};
```

Ticket → Frontend::Agent::ToolBarModule

Frontend::ToolBarModule###1-Ticket::AgentTicketQueue

Toolbar Item for a shortcut. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2".

This setting is not active by default.

Default value:

```
$Self->{'Frontend::ToolBarModule'}->{'1-Ticket::AgentTicketQueue'} =  {
  'AccessKey' => 'q',
  'Action' => 'AgentTicketQueue',
  'CssClass' => 'QueueView',
  'Icon' => 'fa fa-folder',
  'Link' => 'Action=AgentTicketQueue',
  'Module' => 'Kernel::Output::HTML::ToolBar::Link',
  'Name' => 'Queue view',
  'Priority' => '1010010'
};
```

Frontend::ToolBarModule###2-Ticket::AgentTicketStatus

Toolbar Item for a shortcut. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2".

This setting is not active by default.

Default value:

```
$Self->{'Frontend::ToolBarModule'}->{'2-Ticket::AgentTicketStatus'} =  {
  'AccessKey' => 'S',
  'Action' => 'AgentTicketStatusView',
  'CssClass' => 'StatusView',
  'Icon' => 'fa fa-list-ol',
  'Link' => 'Action=AgentTicketStatusView',
  'Module' => 'Kernel::Output::HTML::ToolBar::Link',
  'Name' => 'Status view',
  'Priority' => '1010020'
};
```

Frontend::ToolBarModule###3-Ticket::AgentTicketEscalation

Toolbar Item for a shortcut. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2".

This setting is not active by default.

Default value:

```
$Self->{'Frontend::ToolBarModule'}->{'3-Ticket::AgentTicketEscalation'} =  {
  'AccessKey' => 'w',
  'Action' => 'AgentTicketEscalationView',
  'CssClass' => 'EscalationView',
  'Icon' => 'fa fa-exclamation',
  'Link' => 'Action=AgentTicketEscalationView',
  'Module' => 'Kernel::Output::HTML::ToolBar::Link',
  'Name' => 'Escalation view',
  'Priority' => '1010030'
};
```

Frontend::ToolBarModule###4-Ticket::AgentTicketPhone

Toolbar Item for a shortcut. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2".

This setting is not active by default.

Default value:

```
$Self->{'Frontend::ToolBarModule'}->{'4-Ticket::AgentTicketPhone'} =  {
  'AccessKey' => '',
  'Action' => 'AgentTicketPhone',
  'CssClass' => 'PhoneTicket',
  'Icon' => 'fa fa-phone',
  'Link' => 'Action=AgentTicketPhone',
  'Module' => 'Kernel::Output::HTML::ToolBar::Link',
  'Name' => 'New phone ticket',
  'Priority' => '1020010'
};
```

Frontend::ToolBarModule###5-Ticket::AgentTicketEmail

Toolbar Item for a shortcut. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2".

This setting is not active by default.

Default value:

```
$Self->{'Frontend::ToolBarModule'}->{'5-Ticket::AgentTicketEmail'} =  {
  'AccessKey' => '',
  'Action' => 'AgentTicketEmail',
  'CssClass' => 'EmailTicket',
  'Icon' => 'fa fa-envelope',
  'Link' => 'Action=AgentTicketEmail',
  'Module' => 'Kernel::Output::HTML::ToolBar::Link',
  'Name' => 'New email ticket',
  'Priority' => '1020020'
};
```

Frontend::ToolBarModule###6-Ticket::AgentTicketProcess

Toolbar Item for a shortcut. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2".

This setting is not active by default.

Default value:

```
$Self->{'Frontend::ToolBarModule'}->{'6-Ticket::AgentTicketProcess'} =  {
  'AccessKey' => '',
  'Action' => 'AgentTicketProcess',
```

```
  'CssClass' => 'ProcessTicket',
  'Icon' => 'fa fa-th-large',
  'Link' => 'Action=AgentTicketProcess',
  'Module' => 'Kernel::Output::HTML::ToolBar::Link',
  'Name' => 'New process ticket',
  'Priority' => '1020030'
};
```

Frontend::ToolBarModule###7-Ticket::TicketResponsible

Agent interface notification module to see the number of tickets an agent is responsible for. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2".

Default value:

```
 $Self->{'Frontend::ToolBarModule'}->{'7-Ticket::TicketResponsible'} =  {
  'AccessKey' => 'r',
  'AccessKeyNew' => '',
  'AccessKeyReached' => '',
  'CssClass' => 'Responsible',
  'CssClassNew' => 'Responsible New',
  'CssClassReached' => 'Responsible Reached',
  'Icon' => 'fa fa-user',
  'IconNew' => 'fa fa-user',
  'IconReached' => 'fa fa-user',
  'Module' => 'Kernel::Output::HTML::ToolBar::TicketResponsible',
  'Priority' => '1030010'
};
```

Frontend::ToolBarModule###8-Ticket::TicketWatcher

Agent interface notification module to see the number of watched tickets. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2".

Default value:

```
 $Self->{'Frontend::ToolBarModule'}->{'8-Ticket::TicketWatcher'} =  {
  'AccessKey' => '',
  'AccessKeyNew' => '',
  'AccessKeyReached' => '',
  'CssClass' => 'Watcher',
  'CssClassNew' => 'Watcher New',
  'CssClassReached' => 'Watcher Reached',
  'Icon' => 'fa fa-eye',
  'IconNew' => 'fa fa-eye',
  'IconReached' => 'fa fa-eye',
  'Module' => 'Kernel::Output::HTML::ToolBar::TicketWatcher',
  'Priority' => '1030020'
};
```

Frontend::ToolBarModule###9-Ticket::TicketLocked

Agent interface notification module to see the number of locked tickets. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2".

Default value:

```
 $Self->{'Frontend::ToolBarModule'}->{'9-Ticket::TicketLocked'} =  {
  'AccessKey' => 'k',
  'AccessKeyNew' => '',
  'AccessKeyReached' => '',
  'CssClass' => 'Locked',
  'CssClassNew' => 'Locked New',
  'CssClassReached' => 'Locked Reached',
  'Icon' => 'fa fa-lock',
  'IconNew' => 'fa fa-lock',
  'IconReached' => 'fa fa-lock',
  'Module' => 'Kernel::Output::HTML::ToolBar::TicketLocked',
  'Priority' => '1030030'
```

```
};
```

Frontend::ToolBarModule###10-Ticket::AgentTicketService

Agent interface notification module to see the number of tickets in My Services. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2".

This setting is not active by default.

Default value:

```
$Self->{'Frontend::ToolBarModule'}->{'10-Ticket::AgentTicketService'} =  {
  'CssClass' => 'ServiceView',
  'Icon' => 'fa fa-wrench',
  'Module' => 'Kernel::Output::HTML::ToolBar::TicketService',
  'Priority' => '1030035'
};
```

Frontend::ToolBarModule###11-Ticket::TicketSearchProfile

Agent interface module to access search profiles via nav bar. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2".

This setting is not active by default.

Default value:

```
$Self->{'Frontend::ToolBarModule'}->{'11-Ticket::TicketSearchProfile'} =  {
  'Block' => 'ToolBarSearchProfile',
  'Description' => 'Search template',
  'MaxWidth' => '40',
  'Module' => 'Kernel::Output::HTML::ToolBar::TicketSearchProfile',
  'Name' => 'Search template',
  'Priority' => '1990010'
};
```

Frontend::ToolBarModule###12-Ticket::TicketSearchFulltext

Agent interface module to access fulltext search via nav bar. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2".

This setting is not active by default.

Default value:

```
$Self->{'Frontend::ToolBarModule'}->{'12-Ticket::TicketSearchFulltext'} =  {
  'Block' => 'ToolBarSearchFulltext',
  'CSS' => 'Core.Agent.Toolbar.FulltextSearch.css',
  'Description' => 'Fulltext search',
  'Module' => 'Kernel::Output::HTML::ToolBar::Generic',
  'Name' => 'Fulltext search',
  'Priority' => '1990020',
  'Size' => '10'
};
```

Frontend::ToolBarModule###13-CICSearchCustomerID

Agent interface module to access CIC search via nav bar. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2".

This setting is not active by default.

Default value:

```
$Self->{'Frontend::ToolBarModule'}->{'13-CICSearchCustomerID'} =  {
  'Block' => 'ToolBarCICSearchCustomerID',
```

```
  'CSS' => 'Core.Agent.Toolbar.CICSearch.css',
  'Description' => 'CustomerID search',
  'Module' => 'Kernel::Output::HTML::ToolBar::Generic',
  'Name' => 'CustomerID search',
  'Priority' => '1990030',
  'Size' => '10'
};
```

Frontend::ToolBarModule###14-CICSearchCustomerUser

Agent interface module to access CIC search via nav bar. Additional access control to show or not show this link can be done by using Key "Group" and Content like "rw:group1;move_into:group2".

This setting is not active by default.

Default value:

```
$Self->{'Frontend::ToolBarModule'}->{'14-CICSearchCustomerUser'} =  {
  'Block' => 'ToolBarCICSearchCustomerUser',
  'CSS' => 'Core.Agent.Toolbar.CICSearch.css',
  'Description' => 'Customer user search',
  'Module' => 'Kernel::Output::HTML::ToolBar::Generic',
  'Name' => 'Customer user search',
  'Priority' => '1990040',
  'Size' => '10'
};
```

Ticket → Frontend::Customer

Ticket::Frontend::CustomerDisableCompanyTicketAccess

This option will deny the access to customer company tickets, which are not created by the customer user.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerDisableCompanyTicketAccess'} =  '0';
```

Ticket::Frontend::CustomerTicketOverviewCustomEmptyText

Custom text for the page shown to customers that have no tickets yet (if you need those text translated add them to a custom translation module).

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketOverviewCustomEmptyText'} =  {
  'Button' => 'Create your first ticket',
  'Text' => 'Please click the button below to create your first ticket.',
  'Title' => 'Welcome!'
};
```

Frontend::CustomerUser::Item###15-OpenTickets

Customer item (icon) which shows the open tickets of this customer as info block. Setting CustomerUserLogin to 1 searches for tickets based on login name rather than CustomerID.

Default value:

```
$Self->{'Frontend::CustomerUser::Item'}->{'15-OpenTickets'} =  {
  'Action' => 'AgentTicketSearch',
  'Attributes' => 'StateType=Open;',
  'CSS' => 'Core.Agent.CustomerUser.OpenTicket.css',
  'CSSClassNoOpenTicket' => 'NoOpenTicket',
  'CSSClassOpenTicket' => 'OpenTicket',
  'CustomerUserLogin' => '0',
```

```
  'IconNameNoOpenTicket' => 'fa-check-circle',
  'IconNameOpenTicket' => 'fa-exclamation-circle',
  'Module' => 'Kernel::Output::HTML::CustomerUser::GenericTicket',
  'Subaction' => 'Search',
  'Target' => '_blank',
  'Text' => 'Open tickets (customer)'
};
```

Frontend::CustomerUser::Item###16-OpenTicketsForCustomerUserLogin

Customer item (icon) which shows the open tickets of this customer as info block. Setting CustomerUserLogin to 1 searches for tickets based on login name rather than CustomerID.

This setting is not active by default.

Default value:

```
$Self->{'Frontend::CustomerUser::Item'}->{'16-OpenTicketsForCustomerUserLogin'} =  {
  'Action' => 'AgentTicketSearch',
  'Attributes' => 'StateType=Open;',
  'CSS' => 'Core.Agent.CustomerUser.OpenTicket.css',
  'CSSClassNoOpenTicket' => 'NoOpenTicket',
  'CSSClassOpenTicket' => 'OpenTicket',
  'CustomerUserLogin' => '1',
  'IconNameNoOpenTicket' => 'fa-check-circle',
  'IconNameOpenTicket' => 'fa-exclamation-circle',
  'Module' => 'Kernel::Output::HTML::CustomerUser::GenericTicket',
  'Subaction' => 'Search',
  'Target' => '_blank',
  'Text' => 'Open tickets (customer user)'
};
```

Frontend::CustomerUser::Item###17-ClosedTickets

Customer item (icon) which shows the closed tickets of this customer as info block. Setting CustomerUserLogin to 1 searches for tickets based on login name rather than CustomerID.

This setting is not active by default.

Default value:

```
$Self->{'Frontend::CustomerUser::Item'}->{'17-ClosedTickets'} =  {
  'Action' => 'AgentTicketSearch',
  'Attributes' => 'StateType=Closed;',
  'CSS' => 'Core.Agent.CustomerUser.OpenTicket.css',
  'CSSClassNoOpenTicket' => 'NoOpenTicket',
  'CSSClassOpenTicket' => 'OpenTicket',
  'CustomerUserLogin' => '0',
  'IconNameNoOpenTicket' => 'fa-power-off',
  'IconNameOpenTicket' => 'fa-power-off',
  'Module' => 'Kernel::Output::HTML::CustomerUser::GenericTicket',
  'Subaction' => 'Search',
  'Target' => '_blank',
  'Text' => 'Closed tickets (customer)'
};
```

Frontend::CustomerUser::Item###18-ClosedTicketsForCustomerUserLogin

Customer item (icon) which shows the closed tickets of this customer as info block. Setting CustomerUserLogin to 1 searches for tickets based on login name rather than CustomerID.

This setting is not active by default.

Default value:

```
$Self->{'Frontend::CustomerUser::Item'}->{'18-ClosedTicketsForCustomerUserLogin'} =  {
  'Action' => 'AgentTicketSearch',
  'Attributes' => 'StateType=Closed;',
```

```
  'CSS' => 'Core.Agent.CustomerUser.OpenTicket.css',
  'CSSClassNoOpenTicket' => 'NoOpenTicket',
  'CSSClassOpenTicket' => 'OpenTicket',
  'CustomerUserLogin' => '1',
  'IconNameNoOpenTicket' => 'fa-power-off',
  'IconNameOpenTicket' => 'fa-power-off',
  'Module' => 'Kernel::Output::HTML::CustomerUser::GenericTicket',
  'Subaction' => 'Search',
  'Target' => '_blank',
  'Text' => 'Closed tickets (customer user)'
};
```

CustomerFrontend::CommonParam###Action

Defines the default used Frontend-Module if no Action parameter given in the url on the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerFrontend::CommonParam'}->{'Action'} =  'CustomerTicketOverview';
```

CustomerFrontend::CommonParam###TicketID

Default ticket ID used by the system in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerFrontend::CommonParam'}->{'TicketID'} =  '';
```

Ticket → Frontend::Customer::ModuleMetaHead

CustomerFrontend::HeaderMetaModule###2-TicketSearch

Module to generate html OpenSearch profile for short ticket search in the customer interface.

Default value:

```
$Self->{'CustomerFrontend::HeaderMetaModule'}->{'2-TicketSearch'} =  {
  'Action' => 'CustomerTicketSearch',
  'Module' => 'Kernel::Output::HTML::HeaderMeta::CustomerTicketSearch'
};
```

Ticket → Frontend::Customer::ModuleRegistration

CustomerFrontend::Module###CustomerTicketOverview

Frontend module registration for the customer interface.

Default value:

```
$Self->{'CustomerFrontend::Module'}->{'CustomerTicketOverview'} =  {
  'Description' => 'Overview of customer tickets.',
  'NavBar' => [
    {
      'AccessKey' => 'm',
      'Block' => '',
      'Description' => 'Tickets.',
      'Link' => 'Action=CustomerTicketOverview;Subaction=MyTickets',
      'LinkOption' => '',
      'Name' => 'Tickets',
      'NavBar' => 'Ticket',
      'Prio' => '100',
      'Type' => 'Menu'
    },
    {
      'AccessKey' => '',
```

```
      'Block' => '',
      'Description' => 'My Tickets.',
      'Link' => 'Action=CustomerTicketOverview;Subaction=MyTickets',
      'LinkOption' => '',
      'Name' => 'My Tickets',
      'NavBar' => 'Ticket',
      'Prio' => '110',
      'Type' => 'Submenu'
    },
    {
      'AccessKey' => 'M',
      'Block' => '',
      'Description' => 'Company Tickets.',
      'Link' => 'Action=CustomerTicketOverview;Subaction=CompanyTickets',
      'LinkOption' => '',
      'Name' => 'Company Tickets',
      'NavBar' => 'Ticket',
      'Prio' => '120',
      'Type' => 'Submenu'
    }
  ],
  'NavBarName' => 'Ticket',
  'Title' => 'Overview'
};
```

CustomerFrontend::Module###CustomerTicketMessage

Frontend module registration for the customer interface.

Default value:

```
$Self->{'CustomerFrontend::Module'}->{'CustomerTicketMessage'} =  {
  'Description' => 'Create tickets.',
  'NavBar' => [
    {
      'AccessKey' => 'n',
      'Block' => '',
      'Description' => 'Create new Ticket.',
      'Link' => 'Action=CustomerTicketMessage',
      'LinkOption' => '',
      'Name' => 'New Ticket',
      'NavBar' => 'Ticket',
      'Prio' => '100',
      'Type' => 'Submenu'
    }
  ],
  'NavBarName' => 'Ticket',
  'Title' => 'New Ticket'
};
```

CustomerFrontend::Module###CustomerTicketZoom

Frontend module registration for the customer interface.

Default value:

```
$Self->{'CustomerFrontend::Module'}->{'CustomerTicketZoom'} =  {
  'Description' => 'Ticket zoom view.',
  'Loader' => {
    'JavaScript' => [
      'Core.Customer.TicketZoom.js',
      'Core.UI.Popup.js'
    ]
  },
  'NavBarName' => 'Ticket',
  'Title' => 'Zoom'
};
```

CustomerFrontend::Module###CustomerTicketPrint

Frontend module registration for the customer interface.

Default value:

```
$Self->{'CustomerFrontend::Module'}->{'CustomerTicketPrint'} =  {
  'Description' => 'Customer Ticket Print Module.',
  'NavBarName' => '',
  'Title' => 'Print'
};
```

CustomerFrontend::Module###CustomerTicketAttachment
Frontend module registration for the customer interface.

Default value:

```
$Self->{'CustomerFrontend::Module'}->{'CustomerTicketAttachment'} =  {
  'Description' => 'To download attachments.',
  'NavBarName' => '',
  'Title' => ''
};
```

CustomerFrontend::Module###CustomerTicketSearch
Frontend module registration for the customer interface.

Default value:

```
$Self->{'CustomerFrontend::Module'}->{'CustomerTicketSearch'} =  {
  'Description' => 'Customer ticket search.',
  'NavBar' => [
    {
      'AccessKey' => 's',
      'Block' => '',
      'Description' => 'Search.',
      'Link' => 'Action=CustomerTicketSearch',
      'LinkOption' => '',
      'Name' => 'Search',
      'NavBar' => 'Ticket',
      'Prio' => '300',
      'Type' => 'Submenu'
    }
  ],
  'NavBarName' => 'Ticket',
  'Title' => 'Search'
};
```

Ticket → Frontend::Customer::Preferences

CustomerPreferencesGroups###ShownTickets
Defines all the parameters for the ShownTickets object in the customer preferences of the customer interface.

Default value:

```
$Self->{'CustomerPreferencesGroups'}->{'ShownTickets'} =  {
  'Active' => '1',
  'Column' => 'User Profile',
  'Data' => {
    '15' => '15',
    '20' => '20',
    '25' => '25',
    '30' => '30'
  },
  'DataSelected' => '25',
  'Key' => 'Tickets per page',
  'Label' => 'Number of displayed tickets',
  'Module' => 'Kernel::Output::HTML::Preferences::Generic',
  'PrefKey' => 'UserShowTickets',
  'Prio' => '4000'
};
```

CustomerPreferencesGroups###RefreshTime
Defines all the parameters for the RefreshTime object in the customer preferences of the customer interface.

Default value:

```
$Self->{'CustomerPreferencesGroups'}->{'RefreshTime'} =  {
  'Active' => '1',
  'Column' => 'User Profile',
  'Data' => {
    '0' => 'off',
    '10' => '10 minutes',
    '15' => '15 minutes',
    '2' => ' 2 minutes',
    '5' => ' 5 minutes',
    '7' => ' 7 minutes'
  },
  'DataSelected' => '0',
  'Key' => 'Refresh interval',
  'Label' => 'Ticket overview',
  'Module' => 'Kernel::Output::HTML::Preferences::Generic',
  'PrefKey' => 'UserRefreshTime',
  'Prio' => '4000'
};
```

Ticket → Frontend::Customer::Ticket::ViewNew

Ticket::Frontend::CustomerTicketMessage###NextScreenAfterNewTicket
Determines the next screen after new customer ticket in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketMessage'}->{'NextScreenAfterNewTicket'} =
'CustomerTicketOverview';
```

Ticket::Frontend::CustomerTicketMessage###Priority
Allows customers to set the ticket priority in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketMessage'}->{'Priority'} =  '1';
```

Ticket::Frontend::CustomerTicketMessage###PriorityDefault
Defines the default priority of new customer tickets in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketMessage'}->{'PriorityDefault'} =  '3 normal';
```

Ticket::Frontend::CustomerTicketMessage###Queue
Allows customers to set the ticket queue in the customer interface. If this is set to 'No', QueueDefault should be configured.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketMessage'}->{'Queue'} =  '1';
```

Ticket::Frontend::CustomerTicketMessage###QueueDefault
Defines the default queue for new customer tickets in the customer interface.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketMessage'}->{'QueueDefault'} =  'Postmaster';
```

Ticket::Frontend::CustomerTicketMessage###TicketType
Allows customers to set the ticket type in the customer interface. If this is set to 'No', TicketTypeDefault should be configured.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketMessage'}->{'TicketType'} =  '1';
```

Ticket::Frontend::CustomerTicketMessage###TicketTypeDefault
Defines the default ticket type for new customer tickets in the customer interface.

This setting is not active by default.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketMessage'}->{'TicketTypeDefault'} =
'Unclassified';
```

Ticket::Frontend::CustomerTicketMessage###Service
Allows customers to set the ticket service in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketMessage'}->{'Service'} =  '1';
```

Ticket::Frontend::CustomerTicketMessage###SLA
Allows customers to set the ticket SLA in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketMessage'}->{'SLA'} =  '1';
```

Ticket::Frontend::CustomerTicketMessage###ServiceMandatory
Sets if service must be selected by the customer.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketMessage'}->{'ServiceMandatory'} =  '0';
```

Ticket::Frontend::CustomerTicketMessage###SLAMandatory
Sets if SLA must be selected by the customer.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketMessage'}->{'SLAMandatory'} =  '0';
```

Ticket::Frontend::CustomerTicketMessage###StateDefault
Defines the default state of new customer tickets in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketMessage'}->{'StateDefault'} =  'new';
```

Ticket::Frontend::CustomerTicketMessage###ArticleType
Defines the default type for article in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketMessage'}->{'ArticleType'} =  'webrequest';
```

Ticket::Frontend::CustomerTicketMessage###SenderType
Sender type for new tickets from the customer inteface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketMessage'}->{'SenderType'} =  'customer';
```

Ticket::Frontend::CustomerTicketMessage###HistoryType
Defines the default history type in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketMessage'}->{'HistoryType'} =
'WebRequestCustomer';
```

Ticket::Frontend::CustomerTicketMessage###HistoryComment
Comment for new history entries in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketMessage'}->{'HistoryComment'} =  '';
```

CustomerPanelSelectionType
Defines the recipient target of the tickets ("Queue" shows all queues, "SystemAddress" shows only the queues which are assigned to system addresses) in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerPanelSelectionType'} =  'Queue';
```

CustomerPanelSelectionString
Determines the strings that will be shown as recipient (To:) of the ticket in the customer interface. For Queue as CustomerPanelSelectionType, "<Queue>" shows the names of the queues, and for SystemAddress, "<Realname> <<Email>>" shows the name and email of the recipient.

Default value:

```
$Self->{'CustomerPanelSelectionString'} =  '<Queue>';
```

CustomerPanelOwnSelection
Determines which queues will be valid for ticket's recepients in the customer interface.

This setting is not active by default.

Default value:

```
$Self->{'CustomerPanelOwnSelection'} =  {
 'Junk' => 'First Queue',
 'Misc' => 'Second Queue'
```

```
};
```

CustomerPanel::NewTicketQueueSelectionModule
Module for To-selection in new ticket screen in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'CustomerPanel::NewTicketQueueSelectionModule'} =
'Kernel::Output::HTML::CustomerNewTicket::QueueSelectionGeneric';
```

Ticket::Frontend::CustomerTicketMessage###DynamicField
Dynamic fields options shown in the ticket message screen of the customer interface. Possible settings: 0 = Disabled, 1 = Enabled, 2 = Enabled and required. NOTE. If you want to display these fields also in the ticket zoom of the customer interface, you have to enable them in CustomerTicketZoom###DynamicField.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketMessage'}->{'DynamicField'} =  {};
```

Ticket → Frontend::Customer::Ticket::ViewPrint

Ticket::Frontend::CustomerTicketPrint###DynamicField
Dynamic fields shown in the ticket print screen of the customer interface. Possible settings: 0 = Disabled, 1 = Enabled.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketPrint'}->{'DynamicField'} =  {};
```

Ticket → Frontend::Customer::Ticket::ViewSearch

Ticket::CustomerTicketSearch::SearchLimit
Maximum number of tickets to be displayed in the result of a search in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::CustomerTicketSearch::SearchLimit'} =  '5000';
```

Ticket::CustomerTicketSearch::SearchPageShown
Number of tickets to be displayed in each page of a search result in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::CustomerTicketSearch::SearchPageShown'} =  '40';
```

Ticket::CustomerTicketSearch::SortBy::Default
Defines the default ticket attribute for ticket sorting in a ticket search of the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::CustomerTicketSearch::SortBy::Default'} =  'Age';
```

Ticket::CustomerTicketSearch::Order::Default

Defines the default ticket order of a search result in the customer interface. Up: oldest on top. Down: latest on top.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::CustomerTicketSearch::Order::Default'} =  'Down';
```

Ticket::Frontend::CustomerTicketSearch###ExtendedSearchCondition

Allows extended search conditions in ticket search of the customer interface. With this feature you can search e. g. with this kind of conditions like "(key1&&key2)" or "(key1||key2)".

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketSearch'}->{'ExtendedSearchCondition'} =  '1';
```

Customer::TicketSearch::AllServices

If enabled, the customer can search for tickets in all services (regardless what services are assigned to the customer).

This setting can not be deactivated.

Default value:

```
$Self->{'Customer::TicketSearch::AllServices'} =  '0';
```

Ticket::Frontend::CustomerTicketSearch###SearchArticleCSVTree

Exports the whole article tree in search result (it can affect the system performance).

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketSearch'}->{'SearchArticleCSVTree'} =  '0';
```

Ticket::Frontend::CustomerTicketSearch###SearchCSVData

Data used to export the search result in CSV format.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketSearch'}->{'SearchCSVData'} =  [
  'TicketNumber',
  'Age',
  'Created',
  'Closed',
  'State',
  'Priority',
  'Lock',
  'CustomerID',
  'CustomerName',
  'From',
  'Subject'
];
```

Ticket::Frontend::CustomerTicketSearch###DynamicField

Dynamic fields shown in the ticket search screen of the customer interface. Possible settings: 0 = Disabled, 1 = Enabled.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketSearch'}->{'DynamicField'} =  {};
```

Ticket::Frontend::CustomerTicketSearch###SearchOverviewDynamicField
Dynamic fields shown in the ticket search overview results screen of the customer interface. Possible settings: 0 = Disabled, 1 = Enabled.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketSearch'}->{'SearchOverviewDynamicField'} =
{};
```

Ticket::Frontend::CustomerTicketSearch###SearchCSVDynamicField
Dynamic Fields used to export the search result in CSV format.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketSearch'}->{'SearchCSVDynamicField'} =  {};
```

Ticket → Frontend::Customer::Ticket::ViewZoom

Ticket::Frontend::CustomerTicketZoom###NextScreenAfterFollowUp
Determines the next screen after the follow-up screen of a zoomed ticket in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketZoom'}->{'NextScreenAfterFollowUp'} =
'CustomerTicketOverview';
```

Ticket::Frontend::CustomerTicketZoom###ArticleType
Defines the default type of the note in the ticket zoom screen of the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketZoom'}->{'ArticleType'} =  'webrequest';
```

Ticket::Frontend::CustomerTicketZoom###SenderType
Defines the default sender type for tickets in the ticket zoom screen of the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketZoom'}->{'SenderType'} =  'customer';
```

Ticket::Frontend::CustomerTicketZoom###HistoryType
Defines the history type for the ticket zoom action, which gets used for ticket history in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketZoom'}->{'HistoryType'} =  'FollowUp';
```

Ticket::Frontend::CustomerTicketZoom###HistoryComment
Defines the history comment for the ticket zoom action, which gets used for ticket history in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketZoom'}->{'HistoryComment'} =  '';
```

Ticket::Frontend::CustomerTicketZoom###Priority
Allows customers to change the ticket priority in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketZoom'}->{'Priority'} =  '1';
```

Ticket::Frontend::CustomerTicketZoom###PriorityDefault
Defines the default priority of follow-up customer tickets in the ticket zoom screen in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketZoom'}->{'PriorityDefault'} =  '3 normal';
```

Ticket::Frontend::CustomerTicketZoom###State
Allows choosing the next compose state for customer tickets in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketZoom'}->{'State'} =  '1';
```

Ticket::Frontend::CustomerTicketZoom###StateDefault
Defines the default next state for a ticket after customer follow-up in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketZoom'}->{'StateDefault'} =  'open';
```

Ticket::Frontend::CustomerTicketZoom###StateType
Defines the next possible states for customer tickets in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketZoom'}->{'StateType'} =  [
  'open',
  'closed'
];
```

Ticket::Frontend::CustomerTicketZoom###AttributesView
Shows the activated ticket attributes in the customer interface (0 = Disabled and 1 = Enabled).

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketZoom'}->{'AttributesView'} =  {
```

```
  'Owner' => '0',
  'Priority' => '1',
  'Queue' => '1',
  'Responsible' => '0',
  'SLA' => '0',
  'Service' => '0',
  'State' => '1',
  'Type' => '0'
};
```

Ticket::Frontend::CustomerTicketZoom###DynamicField
Dynamic fields shown in the ticket zoom screen of the customer interface. Possible settings: 0 = Disabled, 1 = Enabled.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketZoom'}->{'DynamicField'} =  {};
```

Ticket::Frontend::CustomerTicketZoom###FollowUpDynamicField
Dynamic fields options shown in the ticket reply section in the ticket zoom screen of the customer interface. Possible settings: 0 = Disabled, 1 = Enabled, 2 = Enabled and required.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketZoom'}->{'FollowUpDynamicField'} =  {};
```

Ticket → Frontend::Customer::TicketOverview

Ticket::Frontend::CustomerTicketOverviewSortable
Controls if customers have the ability to sort their tickets.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketOverviewSortable'} =  '';
```

Ticket::Frontend::CustomerTicketOverview###ColumnHeader
Shows either the last customer article's subject or the ticket title in the small format overview.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketOverview'}->{'ColumnHeader'} =  'TicketTitle';
```

Ticket::Frontend::CustomerTicketOverview###Owner
Show the current owner in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketOverview'}->{'Owner'} =  '0';
```

Ticket::Frontend::CustomerTicketOverview###Queue
Show the current queue in the customer interface.

This setting can not be deactivated.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketOverview'}->{'Queue'} =  '0';
```

Ticket::Frontend::CustomerTicketOverview###DynamicField
Dynamic fields shown in the ticket overview screen of the customer interface. Possible settings: 0 = Disabled, 1 = Enabled, 2 = Enabled and required.

Default value:

```
$Self->{'Ticket::Frontend::CustomerTicketOverview'}->{'DynamicField'} =  {};
```

Ticket → Frontend::Queue::Preferences

QueuePreferences###Comment2

Parameters of the example queue attribute Comment2.

This setting is not active by default.

Default value:

```
$Self->{'QueuePreferences'}->{'Comment2'} =  {
  'Block' => 'TextArea',
  'Cols' => '50',
  'Desc' => 'Define the queue comment 2.',
  'Label' => 'Comment2',
  'Module' => 'Kernel::Output::HTML::QueuePreferences::Generic',
  'PrefKey' => 'Comment2',
  'Rows' => '5'
};
```

Ticket → Frontend::SLA::Preferences

SLAPreferences###Comment2

Parameters of the example SLA attribute Comment2.

This setting is not active by default.

Default value:

```
$Self->{'SLAPreferences'}->{'Comment2'} =  {
  'Block' => 'TextArea',
  'Cols' => '50',
  'Desc' => 'Define the sla comment 2.',
  'Label' => 'Comment2',
  'Module' => 'Kernel::Output::HTML::SLAPreferences::Generic',
  'PrefKey' => 'Comment2',
  'Rows' => '5'
};
```

Ticket → Frontend::Service::Preferences

ServicePreferences###Comment2

Parameters of the example service attribute Comment2.

This setting is not active by default.

Default value:

```
$Self->{'ServicePreferences'}->{'Comment2'} =  {
  'Block' => 'TextArea',
  'Cols' => '50',
  'Desc' => 'Define the service comment 2.',
  'Label' => 'Comment2',
  'Module' => 'Kernel::Output::HTML::ServicePreferences::Generic',
  'PrefKey' => 'Comment2',
  'Rows' => '5'
};
```

Appendix C. GNU Free Documentation License

Version 1.1, March 2000

> Copyright (C) 2000 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document, that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters), and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats, suitable for input to text formatters. A copy made in an otherwise Transparent file format, whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include: plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include: PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location, containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously, at no charge,

using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location, until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version, as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License, for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections, in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

. How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.